Doctor Bots - Remote Presence Robots For MDs

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

June 2010

Simple Human Style Hands Within Your Robot's Grasp

- The Robotics Age
  A Look Back At The Future Of Robotics

- Combat Zone
  Dual-Differential RPM Sensing Or A Melty Brain/Translational Drift Robot

- Planning Out Your Software For Microcontrollers

U.S. $5.50 CANADA $7.00

06

0 71486 02422 4

The Evolution Continues

BIOLOID PREMIUM Kit

Authorized kit by ROBO ONE

Possible to build diverse type of robots and program by yourself

Optimized material for robot education class

- CM-510 Main Controller
- RoboPlus
- Gyro Sensor Walk Balancing
- DMS Distance Measurement
- Humanoid Skin
- RC-100 Remote Controller
- Li-Po Battery 11.1V 1000mA
- USB2Dynamixel
- DYNAMIXEL AX-12+ 18 DOF Actuators
- IR Sensor Object Detection

DYNAMIXEL High-performance robot exclusive actuator

ROBOTIS www.robotis.com

AX-12+ / AX-18F (NEW)

	AX-12+	AX-18F
Weight(g) / (oz)	53.5(g) / 1.88(oz)	54.5(g) / 1.92(oz)
Dimension(mm) / (inch)	32×50.1×40(mm) 1.25×1.97×1.57(inch)	32×50.1×40(mm) 1.25×1.97×1.57(inch)
Gear Ratio(material)	1 : 254(enpla)	1 : 254(enpla)
Operation Voltage(V)	9~12	9~12
Holding Torque(kgf·cm)	15 at 12V / 1.5A	18 at 12V / 2.2A
No load speed(RPM)	59	97
Network Interface	TTL	TTL
Position Sensor(Resolution)	Potentiometer(300°/1024)	Potentiometer(300°/1024)
Motor	Cored Motor	Coreless Motor

RX-64

	RX-64
Weight(g) / (oz)	125(g) / 4.4(oz)
Dimension(mm) / (inch)	40.1×61.3×45.8(mm) 1.57×2.41×1.8(inch)
Gear Ratio(material)	1 : 200(metal)
Operation Voltage(V)	12~18.5
Holding Torque(kgf·cm)	52 at 18.5V / 2.6A
No load speed(RPM)	64
Network Interface	RS-485
Position Sensor(Resolution)	Potentiometer(300°/1024)
Motor	Maxon Motor

RX-24F/ RX-28 (NEW)

	RX-24F	RX-28
Weight(g) / (oz)	67(g) / 2.36(oz)	72(g) / 2.53(oz)
Dimension(mm) / (inch)	35.5×50.8×41.8(mm) 1.39×2×1.64(inch)	35.5×50.8×41.8(mm) 1.39×2×1.64(inch)
Gear Ratio(material)	1 : 193(metal)	1 : 193(metal)
Operation Voltage(V)	9~12	12~18.5
Holding Torque(kgf·cm)	26 at 12V / 2.4A	37 at 18.5V / 1.9A
No load speed(RPM)	126	67
Network Interface	RS-485	RS-485
Position Sensor(Resolution)	Potentiometer(300°/1024)	Potentiometer(300°/1024)
Motor	Coreless Motor	Maxon Motor

EX-106+ (NEW)

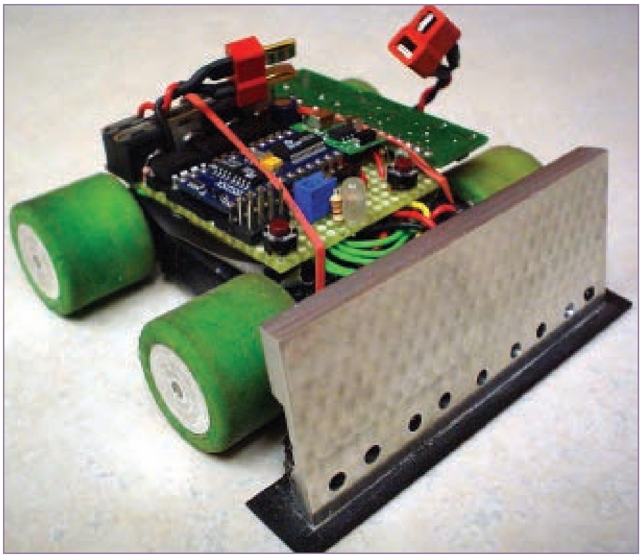
	EX-106+
Weight(g) / (oz)	154(g) / 5.43(oz)
Dimension(mm) / (inch)	40.1×65.3×50.1(mm) 1.57×2.57×1.97(inch)
Gear Ratio(material)	1 : 184(metal)
Operation Voltage(V)	12~18.5
Holding Torque(kgf·cm)	107 at 18.5V / 7A
No load speed(RPM)	91
Network Interface	RS-485
Position Sensor(Resolution)	Magnetic encoder(251°/4096)
Motor	Maxon Motor

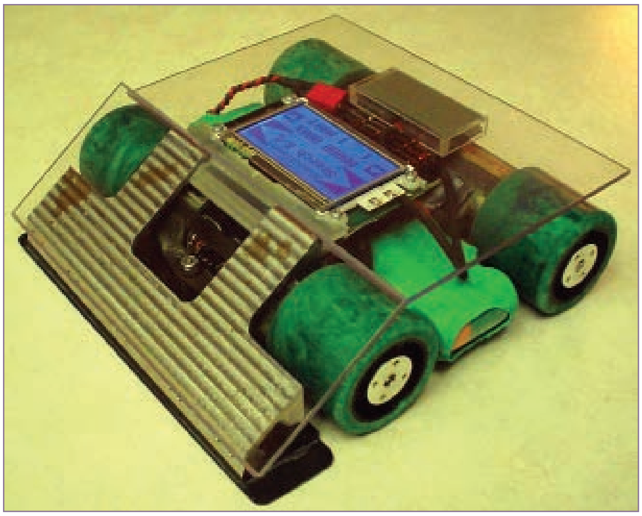
[USA] TEL : 949-333-3635 FAX : 949-333-3634 [KOREA] TEL : +82-70-8671-2600 FAX : +82-70-8230-1336 E-mail : contactus2@robotis.com

SERVO MAGAZINE

06.2010

VOL. 8 NO. 6





The Combat Zone...

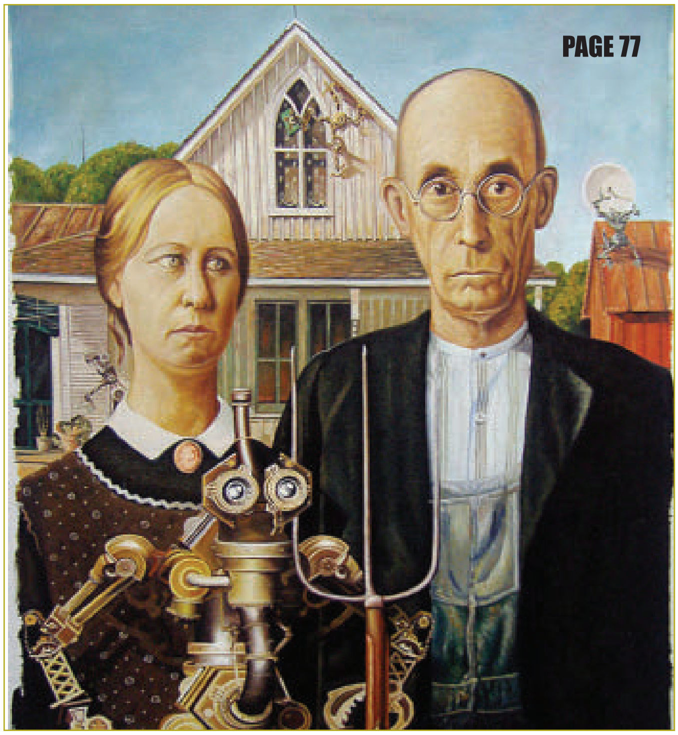
Features

Events

Departments

SERVO Magazine **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...



Columns

Mind / Iron

by Bryan Bergeron, Editor

Mechatronic Music

Jazz guitarist and composer, Pat Matheny, is probably known to most of you. What you may not be aware of is the array of mechatronic instruments behind and around him. If you take a good look at the **photo**, you can see solenoids attached to most of the instruments. The only unmodified, manually operated instrument in the photo is his Ibanez hollowbody guitar.

What you also may not be aware of is Eric Singer — the man responsible for creating all those instruments. Eric is the founder of the non-profit League of Electronic Musical Urban Robots (LEMUR) — a group of artists and technologists who develop robotic musical instruments. He was kind enough to share his background in robotics and music, and how he was able to fuse the two. What follows is a condensed version of my interview with Eric.

Who are you, and what kindled your interest in mechatronic music?

"I list my titles as engineer, musician, programmer, and artist. I have been playing music (mostly saxophones) most of my life. I became interested in computers at an early age, convincing my father to buy us an Apple II. In engineering school, I did some early electronic musical instrument building. Eventually, I saw this as a way of combining my left and right brain interests and as a potential career path. For many years, I built unusual instruments for humans to play computerized sounds — instruments like the Sonic Banana (a bend-sensing rubber tube) and various data gloves and electronic batons. These send musical data to a computer in the form of MIDI (Musical Instrument Digital Interface) messages to produce synthesized music.

At a certain point, I thought, "What would be the flip side of this?" To me, it was to send data in the other direction — out of the computer — and use it to play "real" (i.e., live acoustically based) instruments. This led to the idea of robotic and mechatronic instruments."

What came first for you — your interest in music or mechatronics?

"Music, certainly. I've been playing music professionally for most of my life. In parallel, I was deeply interested in technology — electronics, computers, etc. — from an early age. In college, I began to find ways to combine these interests."

Which of your instruments are you most proud of? Why? How long did it take to create?

"The GuitarBot was the first robot I created with two other members of the group, Kevin Larke and David Bianciardi. It took two years to complete, mostly because we started with no mechanical engineering, robotic, or machining experience. We gained this along the way through experimentation, and trial and error. When we needed to design a mechanical system, we looked to existing devices for inspiration. For example, the sliding bridge mechanism that controls pitch on the GuitarBot is

Photo courtesy of Jimmy Katz.

FOR THE ROBOT INNOVATOR
SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert, Tom Carroll, David Geer, Dennis Clark, R. Steven Rainwater, Fred Eady, Kevin Berry, Fernando Garcia, Matt Rusnak, Jurgen Schmidt, Daniel Ramirez, Tony Cunningham, Pete Smith, Rob Glidden, Dan Toborowski

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

PRODUCTION/GRAPHICS
Shannon Christensen

based on a scanner or inkjet printer head. Throughout 2009, I completely redesigned the GuitarBot to create Model 2.0. Its design is informed by the six intervening years of robot building experience. It is more roadworthy, maintainable, quieter, and reproducible compared to the original, as well as having additional features."

How are your robots different from the first water-powered automatic organs from ninth century Baghdad?

"They are a continuation of a long tradition of people's fascination with and creation of automated musical instruments. Obviously, ours take different forms than these and use electricity instead of water power. But the broad concept is the same.

Automated instruments open up a world of new musical possibilities, whether they are water organs, player pianos, computer-driven synthesizers, or LEMUR robots. They allow humans to create music in ways that are unconstrained by human playing ability. This is not to say that robots and automated instruments are better or worse than human musicians — just that they are different, can be used in different ways, and produce different results. I believe that any technology — analog, digital, mechanical, or other — that opens up new musical possibilities is a good thing."

How long does it take for you to build a music robot?

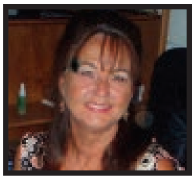
"This is highly dependent on the complexity of the instrument. Some of our mechanisms — such as the ModBots (modular percussion robots) — are specifically designed to be as simple and reproducible as possible. In a day, I can crank out a few dozen drum beaters and turn them into instruments. I've already created driver boxes that control up to 30 ModBots of various varieties (beaters, shakers, scrapers, etc.). Then, it's just a matter of plugging them in and configuring the box through menu-based software created in the application "Max"."

What can you tell me about your robot guitars?

Each unit generally consists of four independent, single-stringed electric slide guitar-like units (four is an arbitrary but useful number of strings ... could be more or less). Each unit consists of a 3" x 36" aluminum plate with robotics, an electric guitar string, picking and damping mechanisms, and a custom microprocessor-based control board. Each unit converts MIDI messages into the signals necessary to control the robotics and play the requested notes.

For musicians and composers, their process is much the same as working with synthesizers or samplers. They simply send MIDI data, and the firmware on the microprocessors takes care of all the details to convert this into music from the instruments." "One of my philosophies for the group has been to get the instruments out into the world and utilize them in as many contexts as possible. To this end, we play concerts with a diverse roster of musicians who compose for and perform live with the robots; we do interactive installations in museums; we build custom instruments for other musicians — Pat Matheny commissioned over 40 LEMUR instruments and mounted a world tour with them in 2010 — and we collaborate with artists of all disciplines — music, visual, dance, theater, etc. — to create new works for the instruments."

I hope this interview with Eric stimulates you to think of how you could apply your knowledge of mechatronics to your other interests, whether that includes music, photography, kite flying, or other seemingly unrelated activities. Eric certainly has me thinking of robotics with a new spin.
To learn more about LEMUR, check out **lemurbots.org**, where there is an extensive video library of instruments and past performances. **SV**

Robytes

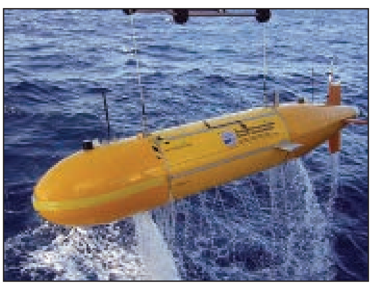
by Jeff and Jenn Eckert

World's Finest TV Sidekick Bot

"Geoff Peterson," Craig Ferguson's robotic sidekick on "The Late Show."

If you don't stay up late, you may not have seen it, and even if you do, the novelty may have worn off by the time you read this, but CBS introduced TV's first robotic second banana back in April. Appearing as Craig Ferguson's mechanical skeleton sidekick, it is named "Geoff Peterson" for reasons that are more inscrutable than funny. Geoff isn't all that versatile, but he can speak seven phrases, move his jaw, turn his head, and raise one arm. Plus, his eyes light up which is more than Ed McMahon could do. The hostbot was built by Grant Imahara of Mythbusters fame (**dsc.discovery.com/tv/mythbusters**). It's worth noting that this is not Imahara's first venture into ghastly mechatronic devices. Apparently, he is largely responsible for the Energizer bunny, as well. As of this writing, you can still view the introductory episode at **www.cbs.com/late_night/late_late_show**.

1.08 Leagues Under the Sea



An Autosub6000 being readied for launch from the Royal Research Ship James Cook.

The folks at Britain's National Oceanography Centre (**www.noc.soton.ac.uk**) have been sending AUVs into the ocean depths for a few years now, but they recently announced the most ambitious adventure yet. The plan is to send an Autosub6000 AUV more than three miles down into the Caribbean to seek the deepest "black smoker" vents and poke around inside. Even though Captain Nemo hung out at almost four times that depth, this is no small feat, given that black smokers are undersea volcanic springs that spew mineral-rich water at temperatures hot enough to melt lead (about 620°F).

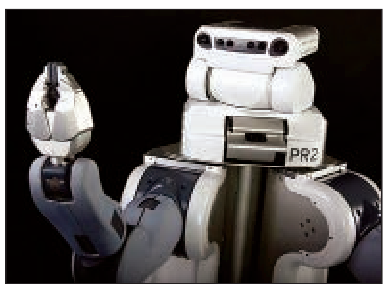
Apparently, the vents "support lush colonies of deep sea creatures that thrive in the otherwise sparsely populated abyss," and the researchers intend to find out more about them. The expedition includes a deep sea vehicle called HyBIS which can be remotely controlled from the surface ship to film the ocean floor, and collect samples of rocks and wiggly little creatures. The researchers will also leave instruments on the ocean floor to monitor currents and deploy experiments to investigate how deep sea creatures colonize new habitats. If all goes according to plan, the voyage will end on April 24, so you should be able to view some of the results at **www.thesearethevoyages.net**.

Return of the Geminoid

Geminoid-F's inspirational model touches her own cheek.

Several years ago, Hiroshi Ishiguro, a professor at Osaka University (**www.osaka-u.ac.jp/en**), creeped out the world by building Geminoid HI-1: a robot that looked exactly like him and imitated Ishiguro's own movements. Well, he's back with Geminoid-F — a female version based on an unnamed woman in her twenties. In a demonstration, the bot smiled and furrowed its brow to mimic the real thing, using programming based on a video of the model. "I felt like I had a twin sister," the woman later told reporters. The F model is designed to work in places like hospitals where it gives patients under examination "psychological security via comforting smiles." The new model uses just 12 actuators versus the HI-1's 46, and built-in air valves that power some of its movements. You can pick one up for a mere $110,000.

Towel Boy Automaton



Willow Garage's PR2, modified at Berkeley to fold towels.

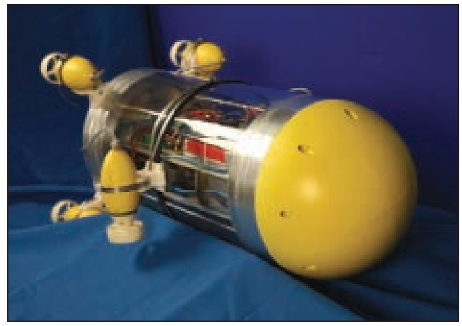
It seems unlikely to generate much excitement anywhere beyond steam rooms and pool cabanas, but apparently teaching a robot to fold towels is quite an achieved. This has been accomplished by a team at US Berkeley's Department of Electrical Engineering and Computer Sciences (**www.eecs.berkeley.edu**) using a PR2 bot from Willow Garage (**www.willowgarage.com**). According to the team, "the task involves one that's proved a challenge for robots: perceiving and manipulating deformable objects —

things that are flexible, not rigid, so their shape isn't predictable."

In operation, the robot "picks one up and turns it slowly, first with one arm and then with the other. It uses a pair of high resolution cameras to scan the towel to estimate its shape. Once it finds two adjacent corners, it can start folding. On a flat surface, it completes the folds — smoothing the towel after each fold and making a neat stack." I'm still not ready to bring out the hats and horns, but I have to admit that folding towels neatly seems to be beyond my own capabilities. If you want details, you can download the full report at **www.jkeckert.com/free downloads/towels.pdf**.

UAV Navigates Murky Water

Also designed for navigating difficult aquatic environments is "Snookie," developed by researchers at the Technische Universität München within the framework of the CoTeSys (Cognition for Technical Systems) organization (**www.cotesys.de**). The Plexiglas and aluminum AUV — measuring 80 x 30 cm and driven by six propeller gondolas — employs sensors that are based on a blind Mexican cave fish (called Astyanax), which navigates via a "lateral-line" organ found only in fish and some amphibians.



"Snookie," the robotic blind Mexican cave fish.

Using this organ, the fish perceive minute variations in pressure and current flow. Even in murky water, this allows them to form a detailed picture of their immediate surroundings at a range of about one body length. According to the researchers, these sensors are comparable to the human inner ear where fine sensory hairs enable us to distinguish sounds ranging from a whisper to the 1812 Overture. No specific projects or implementations have been announced, but the CoTeSys folks "expect such capabilities to enable underwater robots to work autonomously in operations ranging from deep sea exploration to inspection of sewer pipes." **SV**

GEER HEAD

by David Geer

Contact the author at geercom@windstream.net

Remote Presence Robots Put Physicians Within Reach

The Remote Presence (RP) robots from InTouch Health enable physicians to be literally anywhere — in the emergency room, the ICU, at home, the office, or even at the same or different hospital. The physician can see and hear the patient, the attending staff, and the medical equipment in use such as electronic stethoscopes, otoscopes, ECG machines, ultrasound, etc. The patient and staff can see and hear the doctor as he participates in the diagnosis and treatment. The physician can operate many robots in many locations from any one of several control stations. Not only can doctors be where they can't physcially be, but they can be in several places (almost) at once.

Remote Presence Robot Technology

The first RP robot — the RP-7 — is a five foot tall mobile robot that rolls on three balls, six inches in diameter. The balls are each capable of passive or driven movement, so it can move in any direction and change course instantly. The drive system enables the robot to rotate as it moves.

Sensors around the base of the robot let it move freely in crowded rooms without collisions. The physician is informed via a grid of green, red, and yellow squares on a display that represent data transmitted from the sensors. These squares demonstrate to the doctor visually how close he or she is to surrounding objects so they can be maneuvered past.

"This is very valuable in constrained environments as the robot can move directly sideways and avoid obstacles. The robot maps directly to how a human being would move and how the joystick at the control station moves when under the doctor's control," says

A physician is seated at the control station that enables him to teleoperate the robot, as well as to see and hear the patient and staff at the hospital. The wide-angle display presents a view of the robot's environment for navigation and to examine patients and converse with staff. On the right, brain scan results are visible.

The doctor uses a joystick to navigate and a keyboard for text input. A camera and speakers sit atop the display. A PC and speakers are also visible, which process the data the doctor is working with and enable the doctor to hear what is going on at the patient's location, respectively.

Timothy Wright, Vice-President of Marketing and Steve Jordan, Executive Vice-President of Research & Development, InTouch Health.

The robot's display head can pan/tilt like a human head, giving the robot a human-like appearance and the doctor a panoramic view of the robot's surroundings.

Prior to RP robots, doctors effected telemedicine via video-conferencing. In this case, the patient had to be brought into the video-conferencing room which was time-consuming, discomforting, and inconvenient.

The mobile RP robot system works anywhere broadband is available and is a great improvement over the immobility of video-conferencing. The robot has antennae that signal to 802.11 wireless networks. The system communicates back and forth using TCP and UDP packets across the TCP/IP network. The technology's data traverses ports 9000 through 9101 which must be open on the firewall. If the robots are behind firewalls, the technology requires that at least the HTTPS port 443 be open.

For the system to function ideally, bandwidth should be 600 kbps or higher in both directions. Network latency or delay should not rise above 300 ms. Packet loss must be minimal. The technology encrypts data using RSA and 128-bit AES encryption. Systems use the TrendMicro OfficeScan technology to scan for viruses.

RP Models and Capabilities

Doctors can ambulate to the patient's bedside, talk to nurses, basically behave the same as if they were actually there, notes Wright. This facility is important in emergency rooms where a specialist is required but very few are available, sometimes for entire regions. "The patient may have had a stroke, for example. The ER doctor may not be a stroke expert but can call for the neurologist," explains Wright. The neurologist can log in to a robot remotely, see the patient, and effect clinical intervention immediately from a distance.

InTouch Health has several iterations of the RP robot. The RP Lite is the same as the standard model, but a nurse must push it into the patient's room. This model is less expensive and has a somewhat lower level of interaction due to its lack of self-locomotion. "If the setting is a small hospital where they don't anticipate the doctor having to move around much, this robot will suffice," says Wright.

There is also an RP robot alternative that is mounted on a boom so that it can be moved into place over a patient during surgery. Another model is cart-mounted for procedure rooms.

The technology works over the Internet and the hospital's existing network infrastructure so that all the doctor really needs from InTouch Health is the company's hardware and software. Additionally, the company has placed special servers out on the Internet backbone to help broker connections between the site the doctor is at and the site where the robot is. The servers enable a peer-to-peer connection. This is especially helpful if either or both sites are behind firewalls.

Image Courtesy of InTouch Health

This is one of the telemedicine robots with the doctor's image viewable in the screen, and the camera and microphone at top to transmit images and sounds for the doctor to see and hear. A speaker in the body of the robot projects the doctor's voice to the remote location.

Image Courtesy of InTouch Health

This robot has to be pushed around by a staff member, so is a lower cost model than the standard telemedicine robot.

This is the RP Lite with a nurse operator.

The RP technology also addresses shifts in the amount of bandwidth that is available to the application. Embedded in the software in both the control stations and the robots is a technology that manages the available bandwidth.

"Our video software will dynamically adjust given the available network. It reacts to the existing QoS to maximize the technology's use of the network quality available," according to Wright. This technology includes some compression and other techniques to deliver real-time video, despite any dips in the broadband speed.

Resources

The InTouch Health website
http://intouchhealth.com/

Videos of Remote Presence
http://intouchhealth.com/videos-action.html

Tools such as data collection, chart information, CT scans, and other image access augment the robotic technology, appearing in the control station displays, enabling seamless workflow for the physician. "This way, the physicians don't see being remote as being a barrier," says Jordan.

InTouch Health works regularly with more than 1,000 doctors and 300 hospitals, retrieving feedback on the RP technology, which they can use to improve on future models. The company offers on-site demos to introduce medical staff to the capabilities and value of the robots.

The software on the control station operates on a typical PC version of Windows. The interface displays a list of robots the doctor has permission to access and log in to, according to Wright.

The RP line is the only remote presence technology cleared by the Food and Drug Administration.

Conclusion

Remote Presence technology enables immediacy of care in emergencies and a doctor's presence where it might be otherwise impossible for them to get to in a timely manner. Whether it's for emergency care or the dispatch of regular duties, time is always of the essence in health care. **SV**

This is a representation of how the doctor can be there in an emergency even when traveling to the hospital himself in time is not possible.

Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Dennis Clark

I've received a ton of questions in the last month or so and I want to try an experiment to get you folks going on your projects. Since there is only so much time in the day, I'm going to try to find some of your answers with "off the shelf" solutions if at all possible. We'll see how this works, but let me know if you prefer more DIY solutions for the future. With that in mind, let's get started!

Q. I am looking for a simple circuit to drive a hobby servo and want to be able to set endpoints on either side of center for the servo to stop at.

I am going to use this to move a lens barrel (focus) on a TV camera lens, so it will have a gear on it. The barrel has mechanical stops so I need to be able to set endpoints with this servo. Thanks for your help as I have searched and searched, and found stuff thats close, but no cigar.

— Erik, Waterbury, CT

A. Erik, I've worked on this for a bit and it all depends on what you want to spend. I'm sensing that you'd like something that you either can build easily or can assemble from a product that you can buy. I have not found anything that you can use that will allow you to set endpoints using a simple analog pot and another pot for the servo position. Creating your own firmware solution that allows you to read three pots can be done, but getting the resolution that I think you will need will be challenging.

There are two parts of this that could use solving: one is how to control the hobby servo; and the other is how to move the gear on the lens. Let's look at the servo controller first. To get your own DIY project going, I would suggest that you get a serial servo controller — one that will allow you to set endpoints and control the movement speed of your servo. It used to be that such controllers would be very spendy indeed — not so any more! Take a look at the Pololu Serial Servo Controller ROB-08897 at **www.sparkfun.com**. This little guy (see **Figure 1**) is inexpensive (under $18 US) and can be interfaced using a simple serial protocol. But what will it talk to? How about the easy to use and easy to buy (read inexpensive) Arduino? SparkFun sells a variety of Arduino boards, but you can shop around and find your own favorite form-factor and price range. Arduino's are often found for less than $20. A SparkFun Arduino board is shown in **Figure 2**. The SparkFun website has some comments attached to this product that points you to source code examples for programming this board. You can work from these examples and add potentiometers to the board to set your position and set hard stop limits for your servo. Program your Arduino, wire your pots to the board, and mount the whole thing into a project box.

Figure 1. Serial servo board.

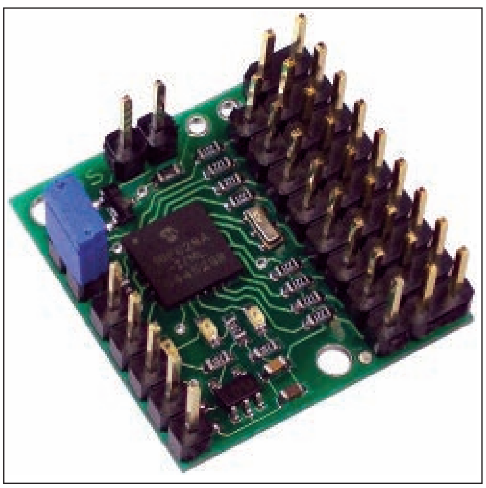
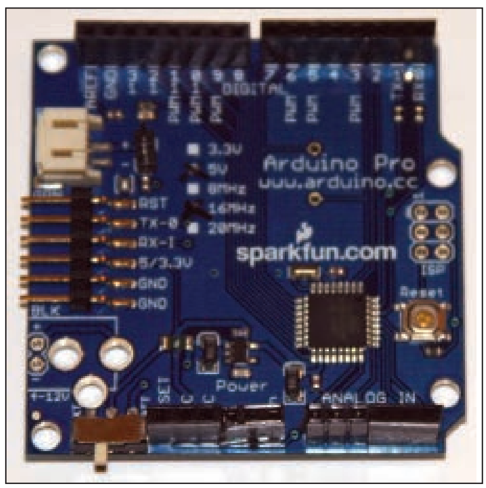


Figure 2. Example Arduino board.



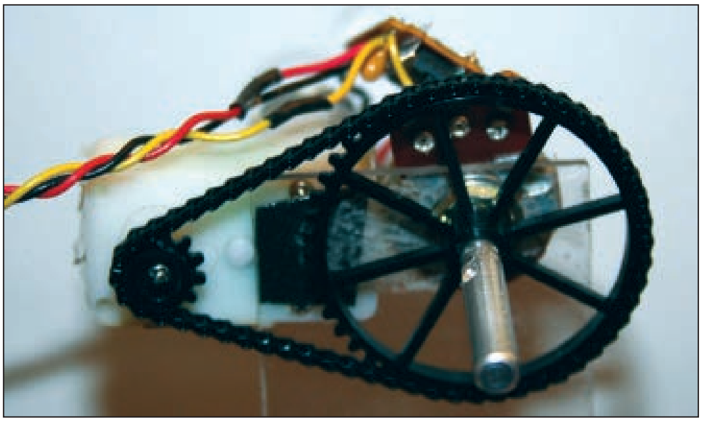


Figure 3. Hacking hobby servo mechanics.

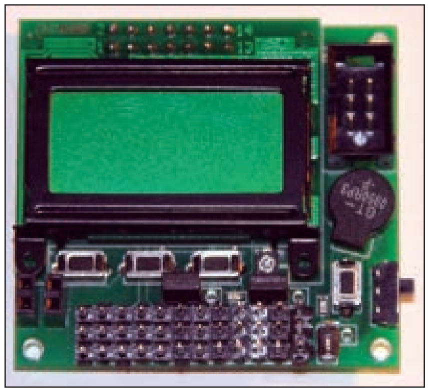


Figure 4. Pololu Orangutan microcontroller.

Part two of this project involves how to make the connection to your lens hardware. Initially, I would suggest taking your servo apart, and using the internal electronics to drive an external motor that was attached to a 5K pot that would replace the servo's internal pot and servo motor. In **Figure 3**, I show just such a setup where I'm using a large *RadioShack* panel mounted pot and a *Solarbotics* (**www.solarbotics.com**) DC motor all hacked to the controller board from inside a *HiTec* (**www.hitecrcd.com**) hobby servo. This will work, and it will allow you to find a motor and gear that could match your lens hardware. If your lens was harder to turn than any hobby servo that you could afford, this is a good option to use. Good, but difficult to implement.

An alternative would be to use the hobby servo "as-is" and hack a drive system onto your lens that will match up with the servo. The photo in **Figure 3** gives a hint about how to do this. Check out the website **www.servocity.com** and look at their selection of chain drive systems that can be mounted directly to your servo. Servo City carries some very sophisticated mechanical devices to attach to your hobby servo or hobby servo controller that can handle a variety of functions. You might find a selection of items that will handle that connection without having to hack into anything.

Q. Kinda new to the field of electronics and robotics and such. Just curious if it is possible that there's a servo out there that will push a button momentarily to sound an alarm and then reset itself to function again after a given time. Any specific recommendations?

— Byrd, Manhattan, KS

A. Byrd, There is no such servo *by itself* that will do what you want. However, you can build a project that will. This question is somewhat related to the previous question I answered above. You would like an alarm to sound so you'll want to add a buzzer to your microcontroller design, or you could look for an "all in one" solution. One such solution is the Orangutan SV-328 robot controller which you can find at **www.pololu.com**. This little board has lots of useful options on it like a small LCD display, a beeper, LEDs, and I/O pins that can be used to drive a hobby servo. The web pages on the Pololu site for this board include a link to a library source that users have assembled to use the capabilities of the SV-328 (and other Orangutan boards) to do various common projects, including hobby servos and — of course — the buzzer on the Orangutan board. At $65 US, it is a bit more expensive than your standard Arduino board, but it has a lot more functionality that will allow you to do all that you want to do on a single board. **Figure 4** shows a photo of an earlier Orangutan board that I have and really like.

You can use any brand of hobby servo that you like or may have on hand. To get your project to push a button, use your favorite material to make a "finger" and glue it to a servo horn. I like to use Lexan plastic and hot glue for my jobs. If your project needs to be more robust, then drill holes and use very small screws to permanently mount your "finger" to the servo control horn.

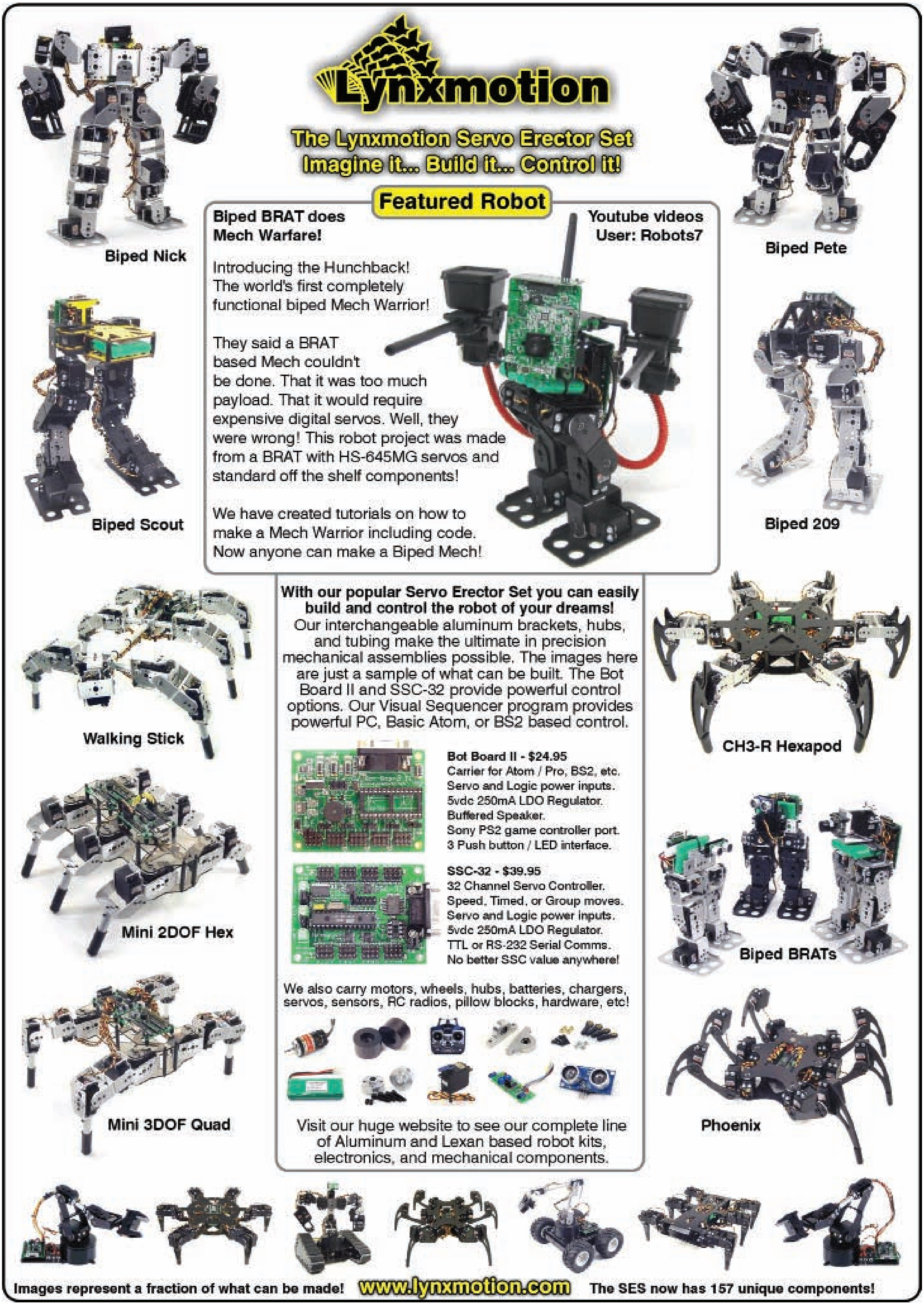
Q. I have a Kondo KHR-1HV Humanoid robot kit. I built it and now want to program it. I have a website and a manual, but I want to take a class or workshop to learn and pass my knowledge onto my students. Any ideas? Classes or schools? It can be anywhere. Hope you can help!

— Jaime, Milwaukee,WI

A. Jaime, I have searched for more information on this robot but have come up blank. It appears that this model is no longer in production and getting help from an English speaking user is going to be difficult.

I don't like to give up though, so I'm putting out a call to all of you helpful *SERVO* readers! If you know of an active user group that can give after-market support to a Kondo KHR-1HV owner, *please* send me an email. This is your chance for 20 minutes of fame! I'll put your information into the next column I write. Any help that you can give will no doubt be appreciated!

Well, that's it for now. I have some good questions queued up for next month dealing with sonar hacking and the Arduino. Keep on building those robots and keep on sending me those questions at roboto@servomagazine.com. I'll do my best to answer them! **SV**


Lynxmotion
The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!
Featured Robot
Biped BRAT does Mech Warfare!
Youtube videos User: Robots7
Introducing the Hunchback! The world's first completely functional biped Mech Warrior!
They said a BRAT based Mech couldn't be done. That it was too much payload. That it would require expensive digital servos. Well, they were wrong! This robot project was made from a BRAT with HS-645MG servos and standard off the shelf components!
We have created tutorials on how to make a Mech Warrior including code. Now anyone can make a Biped Mech!
Biped Nick
Biped Pete
Biped Scout
Biped 209
Walking Stick
CH3-R Hexapod
Mini 2DOF Hex
Biped BRATs
Mini 3DOF Quad
Phoenix
With our popular Servo Erector Set you can easily build and control the robot of your dreams!
Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible. The images here are just a sample of what can be built. The Bot Board II and SSC-32 provide powerful control options. Our Visual Sequencer program provides powerful PC, Basic Atom, or BS2 based control.
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.
SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!
Visit our huge website to see our complete line of Aluminum and Lexan based robot kits, electronics, and mechanical components.
Images represent a fraction of what can be made!
www.lynxmotion.com
The SES now has 157 unique components!

NEW PRODUCTS

INDUSTRIAL ROBOTS

New Mini SCARA Robots

EPSON Robots, introduces the new EPSON G1 Mini SCARA robots. EPSON G1 robots provide an ultra compact yet powerful solution for demanding applications.

"Our goal is to continue our Power of Choice initiative by providing customers with the right robot for the job. While EPSON G1 Mini SCARA robots are small in stature, they are built to the same high EPSON quality and performance standards that our customers have come to know and expect" stated Michael Ferrara, Director of EPSON Robots. "G1 robots are extremely rigid and can handle higher payloads and better duty cycles with faster cycle times vs. competitive models. High rigidity construction combined with superior controls enable G1 robots to achieve industry leading speed and precision which results in more profits for our customers."

G1 Mini SCARA robots are available in arm reaches of 175 or 225 mm. They are also available in Clean and ESD compliant configurations. The 225 mm G1 robot arms can handle many applications with large working range requirements that other robots need 250 mm of reach to handle, thus saving floor space. Unique to the G1 Mini SCARA robots is that both three and four axis models are available. The new three axis models allow for for press fit screw tightening and linear dispensing applications.

With the addition of G1 Mini SCARA robots, EPSON now offers 200+ G-Series SCARA robots, ranging from 175 mm to 1,000 mm in reach, and up to 20 Kg payloads. EPSON G1 robots are ideally suited for applications and industries which require high precision assembly of small components, demanding cycle times, and compact production lines.

For further information, please contact:

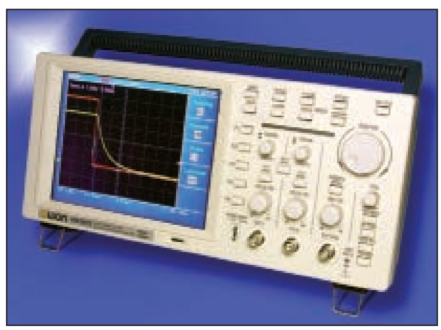
EPSON Robots

Tel: **562•290•5910**
Website: **www.robots.epson.com/products/g-series.htm**

TEST EQUIPMENT

New Features for Low Cost Oscilloscope

Saelig Company, Inc., announces new features for the PDS5022S low-cost, full-featured 25 MHz two-



Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:
newproducts@servomagazine.com

channel benchtop oscilloscope. The new features are: auto-scale, FFT, and trigger hold — normally found on more expensive instruments.

- Autoscale can be set to automatically adjust the vertical gain, the horizontal time base, or both together. This is useful for circuit probing — as the probe is moved from point to point on a circuit board, the display auto-adjusts for best trace presentation. It functions in the same manner as 'AutoSet' but instead of being a one-time function, it's active until turned off, keeping the hands free.
- FFT allows you to instantly see the frequency spectrum of the signal under test.
- Trigger Hold allows you to introduce a delay relative to the trigger point so that a different part of the signal can be seen.

Low cost, two-channel PDS5022S units have useful features normally only seen on higher-end DSOs, such as video trigger, auto-measurements, large 8" full color LCD display, 100 MS/s sampling, XY mode, auto-set, averaging, math functions, USB output, waveform storage, and a three year warranty. A PDS5022S can automatically measure and display frequency and peak-peak/rms/mean values; cursors can also be moved to make individual readings. A built-in self-calibration facility improves measurement accuracy. Video monitoring is also possible since you can trigger on NTSC/PAL/SECAM line or field waveforms. The PDS5022 offers on-board storage and USB output, making this series an ideal choice for design, maintenance, and lab use. PDS5022 features include: manual cursor measurements; up to five automatic measurements; high-speed screen update; storage for up to four waveforms and set-up parameters; 6 KBytes/ch memory; convenient serial interface with software; 400V (DC+AV peak) maximum input; and optional rechargeable battery pack, etc. It has multi-language capabilities, as well.

This lightweight scope is perfect for any engineer's or student's desk. The USB connection makes printing stored results simple. With a large 7.8" 640 x 480 color LCD, PDS5022S is small enough and light enough to carry anywhere. With a buffer size of 6K samples, PDS5022S' sampling range is 10 Sa/s to 100 MSa/s with synchronous eight-bit sampling on both channels. Sensitivity range is 5 mV/div~5V/div with a maximum display of ±50V. Another useful feature is the ability to store four waveforms for comparison with live inputs. A persistence control is available as well, which simulates old analog scopes to compare slow-moving waveforms with previous scans. X-Y display allows you to see phase changes. A+B and A-B math capability is also included.

For further information, please contact:

Saelig Company	Website: **www.saelig.com**

PROTOTYPING BOARDS

Hand Soldering SMT Connectors

SchmartBoard, a company that makes prototyping electronic circuits easier, has announced a new family of boards for prototyping with SMT (Surface-Mount Technology) connectors. These boards support SMT connectors from companies such as Hirose, Molex, Samtec, and Tyco, and pitches of .4 mm, .5 mm, 8 mm, 1.0 mm, and 1.27 mm. They also use the patented SchmartBoard|ez technology which is their solution for hand soldering surface-mount components.

The suggested retail for the boards is $9.99 each or 10 packs for $80.

For further information, please contact:

SchmartBoard	Website: **www.schmartboard.com**

TOOLS & ACCESSORIES

The AccuStar® Electronic Clinometer

Sherborne Sensors, announces the AccuStar® electronic clinometer, designed to provide precision angular

measurements within applications where a high degree of accuracy is required.

The unique design of the AccuStar incorporates a capacitance-based sensor without moving parts. When rotated about its sensitive axis, this sensor provides a fundamentally linear variation in capacitance which is electronically converted into high-precision angular data. Available in a choice of analog, ratiometric, digital or serial models, the AccuStar electric clinometer offers a ±60° measurement range, outputting a signal that corresponds to the direction and magnitude of angular displacement. The highly compact clinometer measures just two inches (50 mm) in diameter and weighs just two ounces (60 grams) for easy installation, with units designed to reliably operate in temperatures of -30 to +65°C. The sensor and its associated low power CMOS electronics are encased in a rugged plastic housing. AccuStar is ideal for a multitude of applications, including off-highway vehicles, wheel alignment, construction equipment, antenna positioning, and robotics applications, as well as other portable or mobile applications where space constraints or environmental concerns preclude the use of other types of precision angle measurement devices.

All AccuStar electronic clinometers are 100% tested and calibrated at the factory prior to shipment, and are accompanied by a two-year comprehensive warranty. The product is also included under the new Sherborne Sensors Guaranteed On-Time Delivery program.

For further information, please contact:

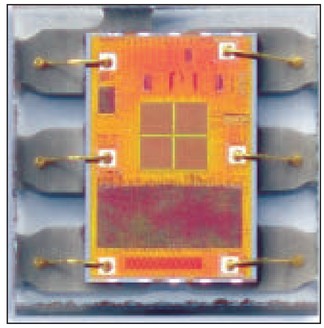
Sherborne Sensors

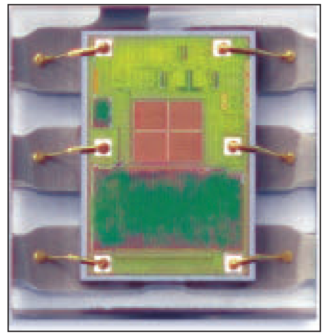
877•486•1766 (US and Canada)
+44 (0) 870 444 0728 (Worldwide)
Email: **sales@sherbornesensors.com**
Website: **www.sherbornesensors.com**

Digital Ambient Light Sensors and Proximity Detectors

Texas Advanced Optoelectronic Solutions®, Inc. (TAOS), is now offering the first members of its next generation digital ambient light sensor (ALS) and proximity detection family that is designed to provide consumer electronics manufacturers with greater freedom to produce sleeker, more appealing, and innovative product designs. These latest TAOS devices eliminate the need to use clear glass/plastic in front of the sensor or drill holes/slots into the display, bezel, or frame in order for light to reach the sensor.

Specifically designed to operate behind darkened glass or other translucent materials, the TSL2771 device family provides both ambient light sensing and proximity detection in a single device that consumes less power than currently available comparable solutions. Proximity detection is ideally suited for touch screen smart phones to automatically turn off the display or control other user functions and can also be used in laptops, desktop computers, and monitors to conserve energy or maintain information privacy when the user is not present. Other emerging applications for proximity sensing technology include touchless display controls activation and management, as well as human gesture detection.





The TSL2771 ambient light sensor and proximity detection device family incorporates an IR LED current-limited driver, analog-to-digital conversion (ADC), interrupt capability, multiple I^2C interface voltage options, and flexible programming allowing it to be configured into many different applications. As darkened glass or translucent materials can be transparent to IR wavelengths while attenuating visible light by 100 times or more, the device utilizes TAOS' patented dual-photodiode architecture that allows the sensor system to compensate for the increased IR component. When combined with the ability to adjust the analog gain settings up to 128X, the device excels in challenging low-light level applications.

"Power consumption is another key advantage to the TSL2771 device family," said Kerry Glover, TAOS, Inc. Applications Manager. "In many of the portable applications, battery life is very critical. Traditional proximity detection methods require a significant amount of power. With the new TAOS architecture, as few as one IR pulse is required for presence detection. This can provide 10x in power savings over traditional analog or other digital solutions."

In applications requiring longer distance detection, an external driver can be utilized to boost the power such that a person can be detected in front of a monitor. A unique state machine allows the device to go into a lower power wait state, thereby only having the high power external detection running a very small portion of the time. For applications requiring only ambient light sensing, this next generation of TAOS devices also includes the TSL2580/81 high-sensitivity ambient light sensors.

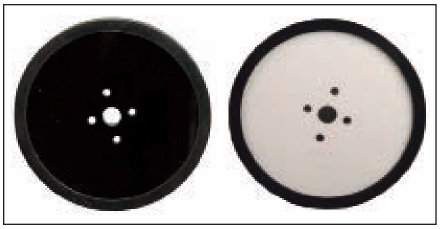
For further information, please contact:

TAOS, Inc.

Tel: **972•673•0759**
Website: **www.taosinc.com**

GIVEAWAYS

Servomotor Wheels



Free plastic 3.12" diameter acrylic wheel(s) are now available from Images with the purchase of a Hitec servomotor. You get one wheel for each servomotor purchased while supplies last. Wheels are available in white, black, and transparent plastic.

The wheels attach to the standard Hitec round servomotor horn (included as a standard part with the servomotor). Four small sheet metal screws are provided to attach the wheels to servomotor horn.

For further information, please contact:

Images Scientific Instruments

Website: **www.imagesco.com/servo/servomotor-wheel.html**

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

JUNE

3-5 **ION Autonomous Lawnmower Competition**
Beavercreek, OH
Autonomous lawn mowing robots compete to see which can mow the grass faster.
www.automow.com

3-5 **University Rover Challenge**
Mars Desert Research Station (MDRS) Hanksville, UT
Teleoperated rovers do sample return, construction, and astronaut rescue events.
www.marssociety.org/urc

4-7 **AUVS International Ground Robotics Competition**
Oakland University, Rochester, MI
Autonomous ground robots must navigate an outdoor obstacle course within a prescribed time while staying within a 5 mph speed limit.
www.igvc.org

5-6 **Motodrone AFO Competition**
Finowfurt, Germany
AFO stands for Autonomous Flying Objects. Robots are judged on their ability to hover in changing wind conditions, fly stably between way points, photograph objects, recover from unexpected dives, and perform precise takeoffs and landings.
www.motodrone.de

6-8 **International Micro Air Vehicle Competition**
Braunschweig, Germany
MAV Surveillance (smallest to complete course wins), MAV Endurance, Ornithopter Competition, and Design Competition.
www.motodrone.de

19-25 **RoboCup Robot Soccer World Cup**
Singapore
If your robot plays soccer, this is the big event of the year. There's soccer simulation, small-sized robot soccer, mid-sized robot soccer, Sony legged robot soccer, RoboCup Junior, Humanoid soccer, and more.
www.robocup.org

24-26 **MATE ROV Competition**
University of Hawaii, Hilo, HI
Remotely operated underwater robots built by student teams compete to complete a series of tasks.
www.marinetech.org/rov_competition/2010

26 **UK National Micromouse Competition**
Birmingham, United Kingdom
Speedy autonomous micromouse robots solve mazes, hoping to win the coveted Brass Cheese award.
www.tic.ac.uk/micromouse

26-27 **International Autonomous Robot Contest**
San Diego County Fairgrounds, San Diego, CA
Events in this competition include Technical Presentation, Urban Challenge, and the Gold Rush Challenge.
www.iaroc.org

26-27 **Robowars Full Metal Carnage**
Queensland, Australia
Remote controlled vehicle combat.
http://qrsc.org.au

30 **Homebrew Robotics Club TABLEBots Phase I Challenge**
CMU, Silicon Valley Campus, Mountain View, CA
In Phase I of the challenge, robots must travel from one side of a table to the other and back without falling off.
www.hbrobotics.org

JULY

7-11 **BOTBALL National Tournament**
Edwardsville, IL
Robots move black and white balls on a game board. A robot of appropriate size is provided in kit form.
www.botball.org

11-15 **AAAI Mobile Robot Competition**
Atlanta, GA
Events include Semantic Robot Vision Challenge, Human-Robot Interaction Challenge, Integration Challenge, and a Robot Exhibition.
www.aaai.org/Conferences/conferences.php

13-18 **AUVS International Underwater Robotics Competition**
Space and Naval Warefare System Center
San Diego, CA
Autonomous underwater vehicles must complete an underwater course with various requirements. Bots cannot be greater than six feet long by three feet wide by two feet deep, and not be greater than 100 kg mass.
www.auvsi.org/competitions/water.cfm

17-20 **RoboBombeiro**
San Miguel Pavilion, Guarda, Portugal
This is a fire-fighting robot contest.
http://robobombeiro.ipg.pt

18-23 **CIG Car Racing Competition**
Barcelona, Spain
This is a competition for race cars controlled by artificially evolved neural nets.
www.wcci2010.org

19-23 **K'NEX K*bot World Championships**
Las Vegas, NV
This event includes Two-wheel drive K*bots (autonomous), Four-wheel drive K*bots (autonomous), Cyber K*bot Division (RC), and Motorless K*bot Division (simple machines).
www.kbotworld.com

24 **RobotRacing**
University of Windsor, Windsor, Ontario, Canada
Autonomous car racing.
www.robotracing.org

AUGUST

6-8 **Rescue Robot Contest**
Kobe, Japan
Teleoperated and autonomous search and rescue.
http://rescue-robot-contest.org

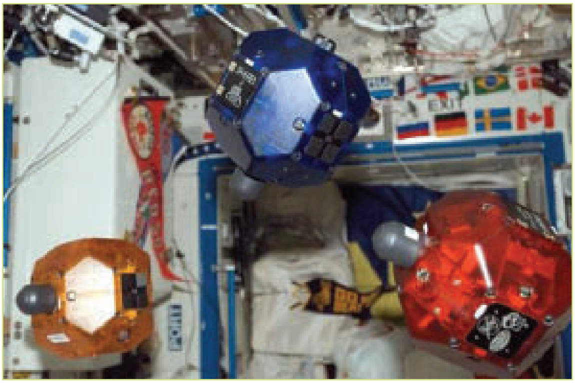
9-13 **AUVS International Aerial Robotics Competition**
University of Puerto Rico at Mayaguez
Puerto Rico
In this event, fully autonomous air vehicles and sub-vehicles perform tasks. There is no size limit, but weight must be under 90 kg/198 lbs (including fuel). This event is open to college students only.
http://avdil.gtri.gatech.edu/AUVS/IARC LaunchPoint.html

SEPTEMBER

3-6 **DragonCon Robot Battles**
Atlanta, GA
Twelve pound max weight single combat; 60 pound max weight single combat. Autonomous and remote control classes. The event is held as part of a science fiction convention called DragonCon.
www.dragoncon.org

5-12 **Microtransat Challenge**
County Kerry, Ireland
Robot boat race.
www.microtransat.org

bots IN BRIEF



ROBOT SPACE BALLS

DARPA has at its disposal millions of dollars to try to make science fiction a reality, and those little SPHERES robots are about to get some pretty sweet upgrades.

First, DARPA wants the SPHERES robots to be able to work outside the comfortable environment of the ISS. They'll have access to their own little airlock, and they'll need to be able to wander around outside semi-autonomously and not get lost. To help them do that, the bots will be able to navigate based on their position and orientation relative to another object which may itself be moving.

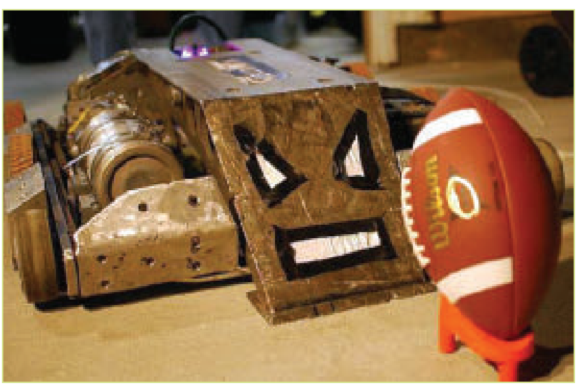
The coolest part, however, is something DARPA calls "Electromagnetic Formation Flight and Power Transfer." The goal of this program is to demonstrate relative stationkeeping, maneuvering, and attitude control between two SPHERES using steerable electromagnetic dipoles at a distance on the order of decimeters to meters. So basically, by manipulating internal electromagnetic coils, the SPHERES robots will be able to attract and repel each other in specific directions, moving without the aid of their CO_2 thrusters. DARPA figures that while they're doing that, they might as well be able to "demonstrate wireless power transfer through resonant inductive coupling." By the way, DARPA will give you a cool $1 million to make it happen. Apply here: **www.fbo.gov**.

KICKIN' IT WITH ZIGGY

Ziggy the Combot and San Francisco 49ers' kicker Joe Nedney took each other on in a place kicking contest recently.

It's worth noting that Ziggy was actually able to kick the ball over 60 yards (repeatedly) in practice, off of a concrete surface with no headwind. The grass surface of the football field was much softer than concrete, meaning that a significant portion of the energy released by the pneumatic arm went to driving the robot down into the ground as opposed to into the football. That's the thing about robots … They work great in the lab, and then all kinds of "stuff" happens when you get them outside.

Suffice it to say that Ziggy will undoubtedly be up for a rematch next year. Meantime, while we humans may not be able to match Ziggy in sheer power, we've still got adaptability going for us. So for now, we're still in charge. Don't forget, however, Ziggy can toss 340 pound robots like they're rag dolls.



bots IN BRIEF

FANTASTIC PHOTO OPPS

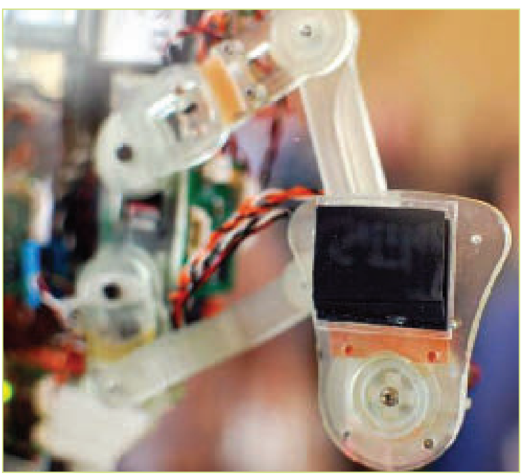
Robots are designed for tasks that are dull, dirty, or dangerous, right? Well, taking close-ups of African wildlife certainly can fall into that last category, especially when you're trying to get super close up and personal with big and temperamental animals (like water buffalo or elephants or lions). Solution? Enter a digital single-lens reflex camera and a cute robot with big wheels. Will and Matt Burrard-Lucas built BeetleCam to get a new perspective on African animals. BeetleBot is controlled remotely from a Range Rover about 50 yards away, and has gotten into quite few shenanigans.

The Burrard-Lucas team thought that elephants would be an easy subject for BeetleCam's first outing. They were wrong because elephants are wary of unfamiliar objects and due to their highly sensitive hearing, they are almost impossible to sneak up on! (And you thought elephants just had good memories.) They soon discovered that the best way to photograph an elephant was to position the camera well in front of it and then let the animal approach it in its own time. With this technique, Will and Matt managed to get some incredible photos of these colossal creatures. Go to **http://blog.burrard-lucas.com/2010/04/adventures-of-beetlecam/** and check them out.

STICKY SITUATION

Meet Stickybot III — a brand new incarnation of Stanford Robotics gecko-inspired climbing robot.

Stickybot III is able to stick to glass by using an adhesive material that mimics gecko feet. Gecko toes are covered in bajillions of microscopic hairs that are so tiny they are attracted to the very molecules in the surface that they're on. The material is not "sticky" in the normal sense but rather the material is attracted to the surface directly through Van der Waals forces.



The difficulty in getting Stickybot III to actually climb is due in large part to the way the sticky pads are attached to its feet. Gecko feet use a bunch of different pliable layers of the sticky hairs which maximizes the surface area of their feet that makes contact with the surface that they're trying to stick to. Stickybot III only has a few layers, so if the feet aren't oriented properly with respect to the surface, a lot of the adhesion is lost. According to Stanford, Stickybot is an embodiment of their hypotheses about the requirements for mobility on vertical surfaces using dry adhesion. The main point is to have a controllable adhesion. Future Spidermen need not apply.

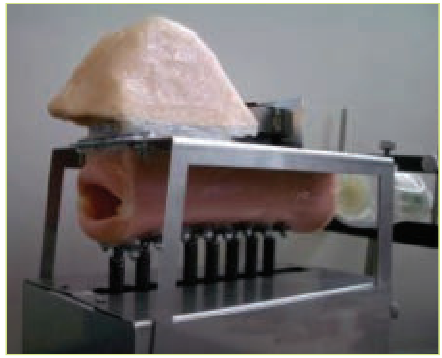
Cool tidbits herein provided by Evan Ackerman at www.botjunkie.com, www.robotsnob.com, www.plasticpals.com, and other places.

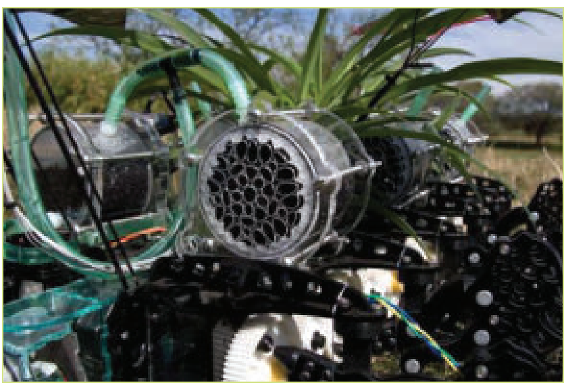
WHAT A CHARACTER

Want your own business mascot? Chararobo (short for Character Robot) will create one for you like the one they made for Cable TV, Inc. Based on Kyuchan (kyu means 9" in Japanese), the 2' 7" bot weighs 8.8 lbs and acts as a greeter at their front desk. When a human approaches, he dances around with his 6° of freedom and Futabo servo motors. You too can get one for $8,600 to $16,000 at **www.chararobo.com**.

LOOK WHAT'S TALKIN'

Japan's Kagawa University engineers built a robot to assist the hearing impaired to speak. Made up of an air pump, a resonance tube, a nasal cavity, and a microphone, the artificial vocal chords talk and use an algorithm to mimic human speech. Once it compares the user's speech to its own, it suggests pronunciation adjustments.





SELF-FEEDING ROBOT PLANT

The Nomadic Plant by Gilberto Esparza is designed to seek out polluted water, clean it up, and feed it needed nutrients. Powered by microbial energy, the art project also feeds itself — proving that robots will save the planet by symbiosis.

This plant-machine hybrid gets its microbial energy from a fuel cell. The microbes at work to generate electricity its robo legs to remain in action do need some nourishment, so when the need arises, the hybrid seeks out water from the polluted Lerma Santiago River in Mexico.

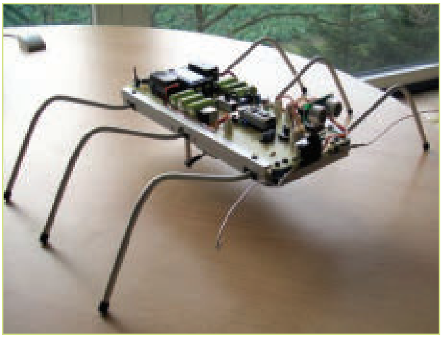
The water is sucked in through a tube that sets the microbes feeding; a portion is also sent to the plant sitting atop the robot legs. Read more at **www.greenpacks.org/2010/04/12/robot-nomadic-plant-feeds-off-polluted-water/#ixzz0mKAy9AOK**.

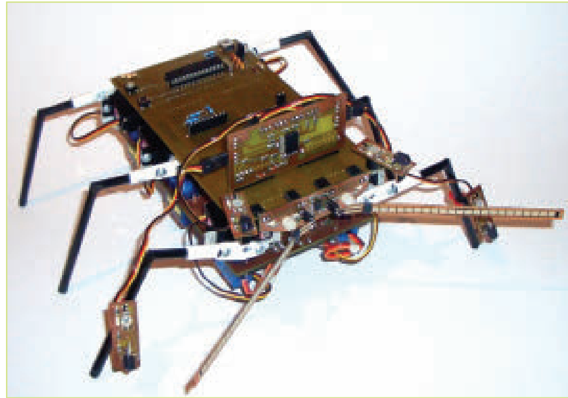
HONEY, I SHRUNK THE ROBOTS

Did you know they were still making Shrinky Dinks, much less that the color-and-bake toys come as robots? The kit comes with 40 pre-cut shapes to make three different bots, eight colored pencils, beads, barrettes, and earring hoops so that your kids can let everyone know you are one cool mom/dad on your next outing.

AMAZING ARACHNIDS

Germany's Steffen Schütte came up with these clever RoboSpiders. The insectobots use a Javelin Stamp for the microcontroller brain and an ultrasonic range sensor for their eyes. The bots can go forward, dodge, escape, and — if all else fails — feign death. RoboSpider III is now appearing at a web near you (**www.steffenschuette.de**).





BLAZING NEW TRAILS

Despite all precautionary measures, firefighting remains a risky task that poses life-threatening challenges to firefighters all over the world. Making their life a bit easier and safer, the Firegard is an autonomous firefighting robot that reduces the exposure of firefighters to intense situations. Capable of operating in different conditions and terrains, the Firegard records important pictures and information via built-in sensors and cameras which help officials assess situations and execute operations without risking the life of the human firefighters. It is still in the concept stage at this point.

ADEPT WITH FOOD HANDLING

We didn't know that the USDA had (or wanted) the power to give a thumbs up to a robot used in food handling but that is just what they did with the Adept Quattro s650HS. The rapidly moving bots have four arm kinematics that make them faster than average. They can be used for meat and poultry processing and are specifically designed for high-speed manufacturing, packaging, material handling, and assembly applications. The design of the Quattro differs from conventional parallel robots in that it features the four-arm kinematic. This unique design enables higher speeds and faster accelerations.

ATTACK BUG

The Kabuto-Mushi (rhinoceros beetle) is a do-it-yourself kit that can attack other robots, small pets, and dirty laundry when completed. It features an infrared remote with eight channels that control four motors — one for the claw position and caterpillar traction. No soldering is needed and all parts are included.

CHARLI IN CHARGE

Dr. Dennis Hong and his legion of robotics students at Virginia Tech's Robotics and Mechanisms Laboratory aren't content to rest on their laurels. Having been the first US team to compete in RoboCup's Humanoid League with DARwIn (already up to its fourth model; see coverage in our Sep '07 issue), they're now ready to unveil CHARLI (Cognitive Humanoid Autonomous Robot with Learning Intelligence) — the first complete, untethered, autonomous full-sized humanoid robot designed and built in the US. And they're building two of them. There's a lightweight version which stands approximately 135 cm (4'5") tall, weighs 12.5 kg (27.5 lbs), and has 20 degrees of freedom. Using what appear to be powerful Dynamixel servos, the team had only $20,000 to acquire the necessary parts and materials. It will compete in the Adult Size Humanoid League at this year's RoboCup competition. Then there's CHARLI-H: a heavier, more sophisticated version that has a completely different set of legs with custom actuators that will allow it to perform even better.

BANKING ON GOOD DIRECTIONS

Say you've got a giant banking center. It has nine buildings and 5,500 employees. That's great, but you can't get any business done, because all 5,500 people get immediately and hopelessly lost as soon as they step outside their cubicles. Solution? Robots of course!

Santander's Group City in Madrid has employed a swarm (their word) of futuristic "interactive guest assistance" robots to help people find their way around the place. And you thought robots couldn't tell you where to go.

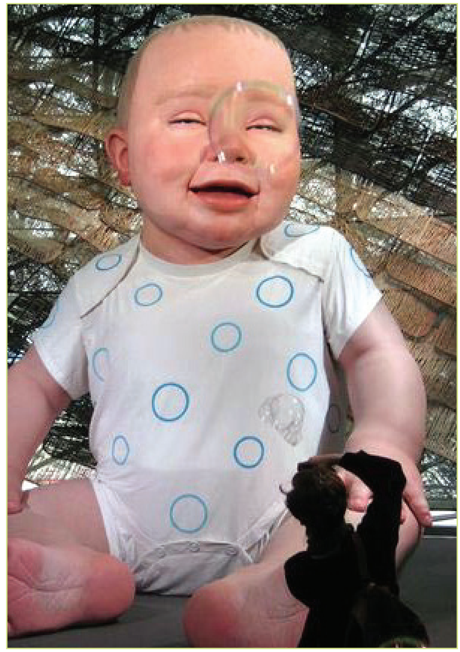
LA, LA, LA, LA, LOLA

Having previously developed a biped named JOHNNIE in 2001, the Technical University of Munich and the Institute of Technology Autonomous Systems (TAS) are developing its successor called LOLA. The robot — which has been in development for six years — is capable of planning its own walking trajectory through a room in real time using image data. Due to the processing power required to perform image recognition, LOLA is connected via cables to three computers which handle this task. Also at Hannover Messe 2010, LOLA was able to detect obstacles in its way such as chairs or people, and modify its path accordingly.

LOLA stands 180 cm (5'10") tall, weighs 60 kg (132 lbs), and has a total of 25 degrees of freedom (two legs x7; two arms x3; waist x2; head x3). While most humanoid robots have only six joints per leg, LOLA has an extra toe joint which reduces load and increases its step length and walking speed. The robot's mechanical structure was carefully designed for optimum strength and efficiency with lightweight components. To help maintain balance, it is equipped with gyro sensors in its upper body and six-axis force sensors in its feet.

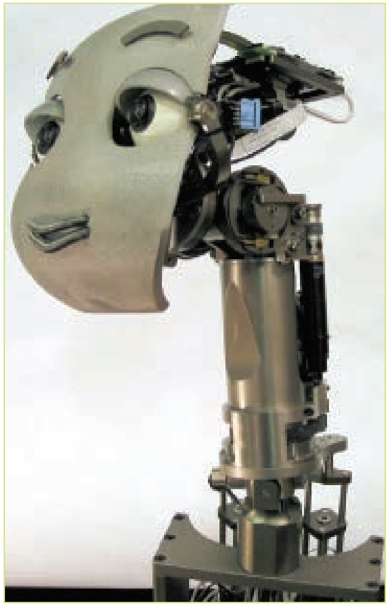
LOLA will soon be capable of running, however, this is a fairly new development. The researchers hope they'll eventually be able to reach a speed of 5 km/h, which is more in line with a human's top walking speed.

BIG BABY



This 6.5 m (21 ft) baby robot — developed by Hollywood visual effects studios behind movies like *Aliens Vs. Predator* and *Star Trek* — was unveiled at the Spanish Pavilion at the Shanghai World Expo that opened to the public May 1st. Its animatronics are reportedly realistic enough to allow the baby to move and giggle, but thankfully not realistic enough to require diaper changes. The exhibit is meant to represent dreams of the future. Maybe Rosemary's dreams ... this is one scary baby!

LET'S FACE IT



Like MIT's Kismet and KIST's FRi, Mertz is an active vision head robot that recognizes and reacts to faces and expressions. It was built in 2004 primarily by Lijin Aryananda and Jeff Weber at the MIT Media Lab. Its main purpose was to research socially situated learning similar to an infant's learning process, so it was programmed to track faces and bright objects, and could repeat the sounds of words extracted from audio data. Unlike other robots that typically don't get out much (interacting mainly with researchers in the lab), Mertz was designed to be able to "live" around people and absorb information for many hours.

Mertz has a total of 13 degrees of freedom, including individually actuated eyebrows and lips, has two cameras for vision, and a microphone. It's about 25 cm (10 inches) tall and weighs 2 kg (4.25 lbs). One of the main issues involved is how to interact while simultaneously learning from a person. Another problem was the ambient noise level of the robot's surroundings. In the lab it was quiet, allowing the robot to more easily parse words, but in the world it was very noisy.

HELP FROM WHAT AILAS YOU

At Hanover Messe 2010 — a trade fair for industrial technology — DFKI Bremen showcased a new humanoid robot called AILA which is demonstrating how robots might be used interactively in dynamic environments with humans by 2020. The system uses SemProM (Semantic Product Memory) which it combines with its computer vision when handling objects of varying shapes and sizes. For example, AILA can adjust how it holds a bottle based on its weight and fragility. Each product or object stores and communicates its properties such as its size, where it needs to be transported, which production line it belongs to, and so on through RFID.

Looking a bit like Toyota's Robina, the feminine AILA is equipped with two laser range finders, stereo vision in its head, a 3D camera for object recognition and orientation, and an RFID reader in its left hand. It has a total of 22 degrees of freedom (wheels x2; two arms x7; torso x4; head x2) and moves on a wheeled base containing six wheels. Thanks to this wheel configuration, AILA can turn in any direction and is stable on smooth to moderately rough surfaces.

COMBAT ZONE

Featured This Month:

PARTS IS PARTS

Data Points: Fine-Tune Your Bot's Performance

● by Pete Smith

Knowing exactly how well your bot performs in a particular configuration would be very useful ... like, exactly how fast is the blade turning? What current is the weapon motor drawing at start-up and at cruise? How much does the battery voltage sag at full power?

Knowing those figures, one could then try different drive and/or weapon motors, bigger or smaller blades, try different gearing to the weapon, higher voltage battery packs, or ones with better current ratings, and see if overall performance or efficiency could be improved.

An easy way to gather that data is with a data logging device such as the Eagletree Micropower E-Logger V3 (**Figure 1**) from **www.eagletreesystems.com**.

These devices (<$100, weight 0.7 oz) connect between the battery and the bot, and directly record battery voltage and current over time. Other sensors can be added to measure motor speed, blade speed, battery and motor temperatures, etc.

I found the most useful additional sensors were those for brushless motor RPM and temperature. The sensors were easy to fit following the instructions supplied.

The E-Logger comes with

FIGURE 1

FIGURE 2

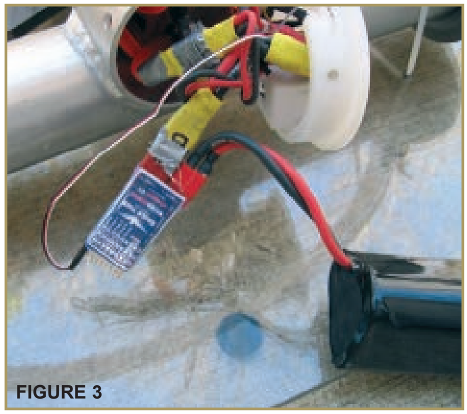


FIGURE 3

software for loading onto your computer. There's a USB cable that allows you to update the code on the E-Logger, then upload data to the PC for display.

To safely test my 12-lber Surgical Strike, I secured it to a large piece of scrap Lexan with tie wraps and had a piece of slippery UHMW screwed down for the blade axle to rotate on (**Figure 2**). I only run it in my backyard because it is very important that there is no chance of injuring anyone if a blade comes loose, for example. With the bot restrained, I can power-up the weapon from a safe distance. Some builders have made small test boxes with a Lexan top cover to allow safe testing. (I would recommend this and plan to build one myself.)

First, you plug the sensors into the E-Logger and then plug it between the battery and the rest of the bot (**Figure 3**). It comes with Deans type connectors so you might need to make adapter cables if you use a different type of connector on your battery.

The logger starts recording as soon as you plug it in. It has a very long recording time (based on the sampling rate), so there is no need to hurry to get the testing done.

Typically, I power-up the blade and let it run, then do a few power off and on cycles to replicate the blade being stopped and restarted in a fight. After three minutes or so (the length of a standard fight), I power-down and make the bot safe. You can then make changes to the bot and repeat the run so you can compare the data.

When you have finished testing, simply unplug the E-Logger, plug it into your PC, and have a look at the results.

I have the software set up to display battery pack volts, current, total watts, weapon motor speed, weapon speed (called "Prop" and is simply the weapon motor speed divided by the gear ratio used), mAH used, and a couple of temperature readings (**Figure 4**). You can play back the recording and watch what happened, and you can also easily graph the results to get the big picture (**Figure 5**).

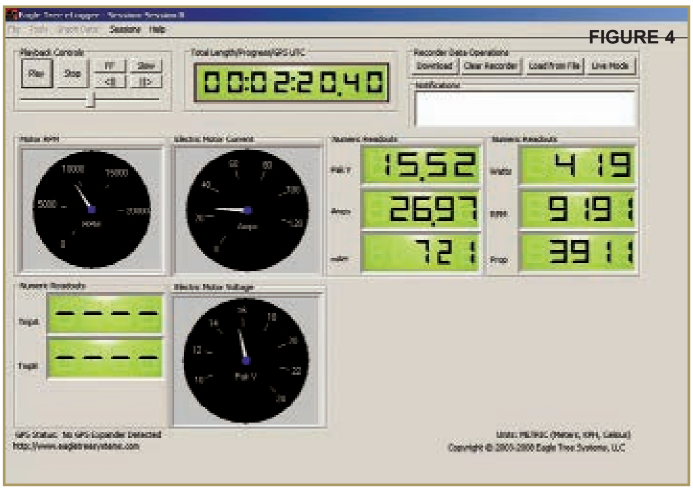


FIGURE 4

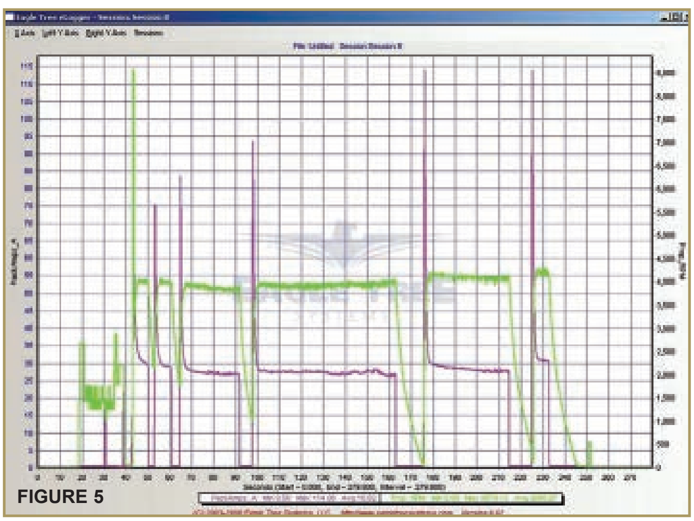


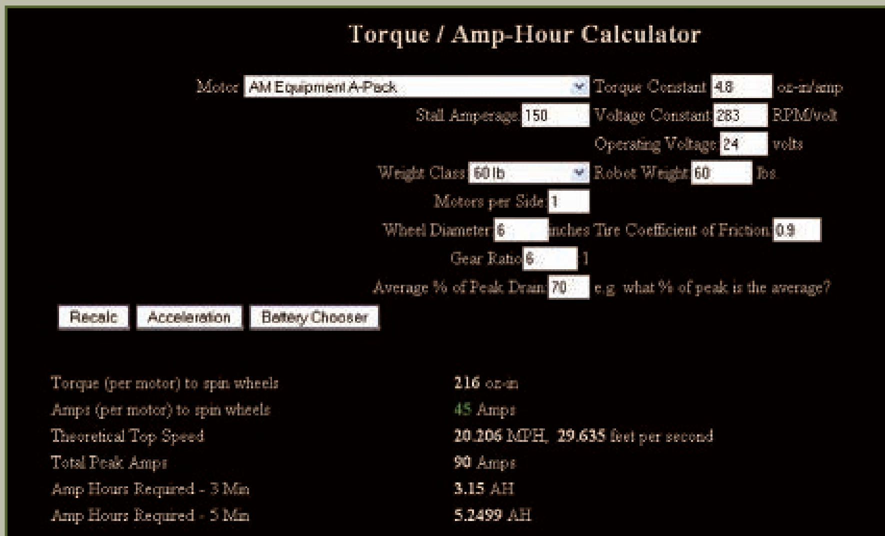
FIGURE 5

Now, you can start making changes to blade design and see if those changes affect blade speed and current draw. Or, add a cell to the battery and see if the added power is worth the weight. You can also check to see if your bot is current-limited by your existing battery by adding two in parallel and seeing if power increases. The possibilities are almost endless.

I have yet to use the E-Logger during an actual fight, but that's next on my to-do list to see if my tests are a reasonable reflection of what happens in the arena. It will be fun to see what the "blade on blade" hits and getting stalled in the corner really does to my bot! **SV**

MANUFACTURING: TENTACLE CALCULATOR REHOSTED BY KILLERBOTICS

● by Kevin M. Berry



The combat robotics community was deeply saddened by the passing of Steve Judd last year. Steve was an icon in the sport, and left behind a lasting legacy in many areas. One of those — his Torque/Amp-Hour Calculator — is a standard tool used by dozens of builders, and one of the first "pointers" usually sent to new builders.

Once again stepping up in support of builders everywhere, Dick Stuplich of Killerbotics volunteered to re-host the calculator, with the gracious permission of Nora Judd — another long-time participant and advocate of combat robotics.

Along with the Tentacle Calculator, Dick also supports a Sprocket Center Distance Calculator, a Chain Length Calculator, and a Bandwidth/Data Type Calculator. This is in addition to BotRank — the "by popular demand" comparison site for bot fighters — which is the data source for the (currently on hiatus) "Top Bots" column here in the Combat Zone.

SERVO thanks Dick, Nora, and — of course — Steve, for their service to the community (**www.killerbotics.com/kbtools/**). SV

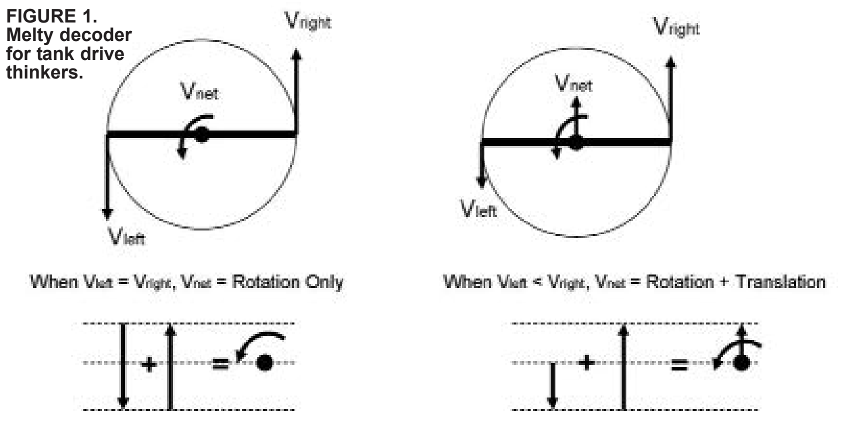
Open Melt — Generous Sharing or a Threat to Society as We Know it?

● by Kevin M. Berry

The author — in a blatant attempt to up his word count and thus his income — summarizes material he's already sold to SERVO once ...

In February and March '08, Combat Zone published a tutorial and "State of the Melt" series on the amazing and unique bot drive system: translational drift. Because bot builders are genetically incapable of using the correct, technical terms for anything, this brain twisting drive methodology quickly became known as "Melty Brain."

FIGURE 1. Melty decoder for tank drive thinkers.



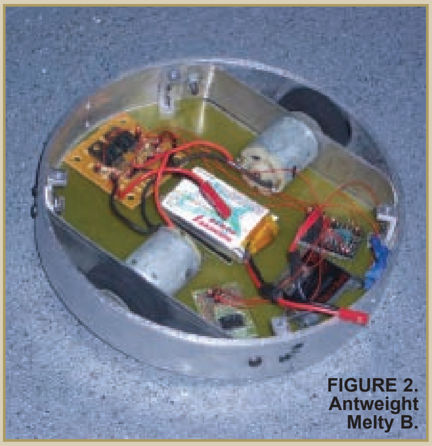


FIGURE 2. Antweight Melty B.

In that series, we looked at four bots that tackled the difficult task of making the whole bot a kinetic energy weapon using this interesting phenomenon of rotating bodies. In a traditional "Full Body Spinner" (FBS) — contrary to the name — the outer shell spins, but the base platform is a traditional "tank drive" vehicle. In a Melty, the whole bot spins, and by varying the drive speed of each wheel as the bot spins, a translational movement results.

Ilya Polyakov of Team Carnivore is generally credited with the first attempt at putting this technology in the box. He says: "At the time, I was taking Dynamics as part of my M.E. course work and the vector math behind combined translational and rotational motion really hit the spot. The identical but opposite rotational velocity vectors combined with a single translational vector made sense in the tank drive perfectly."

Figure 1 attempts to translate (sorry, pun alert) this jargon into language the mere mortal can understand without having the dreaded (and aptly named) brain meltdown.

Rich Olson of Nothing Labs (previously known as Team SpamButcher) has emerged as the new King of Melty. A while back, he built a successful Antweight bot using Melty Brain technology. "To determine its relative position in each spin," Rich says, "Melty B (**Figure 2**) uses an accelerometer to measure the centrifugal force created by the rotation of the bot. The level of G force detected is then run through a formula that accurately determines how fast the robot is spinning. Once the exact spin rate is known, it's possible to determine where it is in the current spin based on timing."

The robot flashes an LED each time it hits a point in the spin it thinks is "forward" which appears as a streak, indicating to the driver which direction the robot will move when the stick on the remote is pushed forward. The bot has a top speed of 1,400 RPM and can "translate" at about 1.5 feet per second. At top speed, forces inside the robot can reach over 100 Gs.

Rich coded his original software — over 500 line's worth — in Bascom AVR. The microcontroller is an Atmega 168 Pololu Baby Orangutan. He used the Bascom AVR compiler

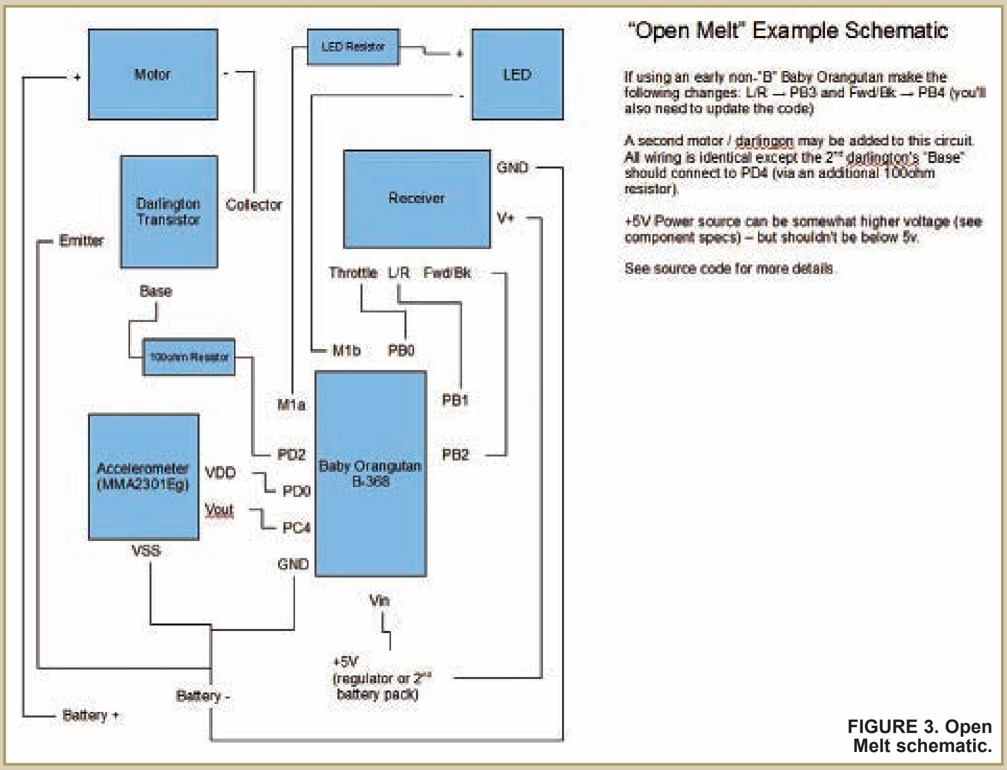


FIGURE 3. Open Melt schematic.

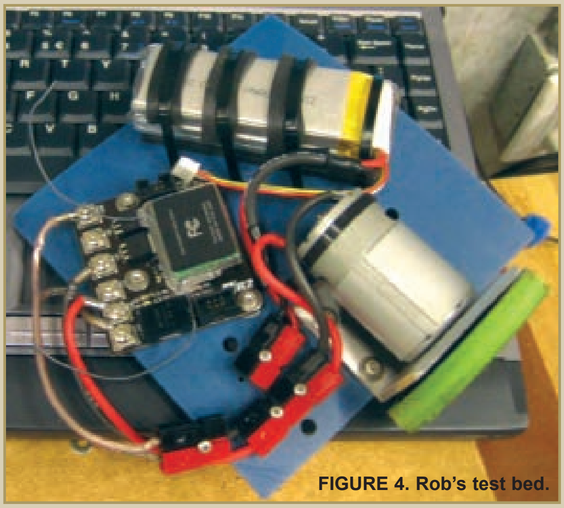


FIGURE 4. Rob's test bed.

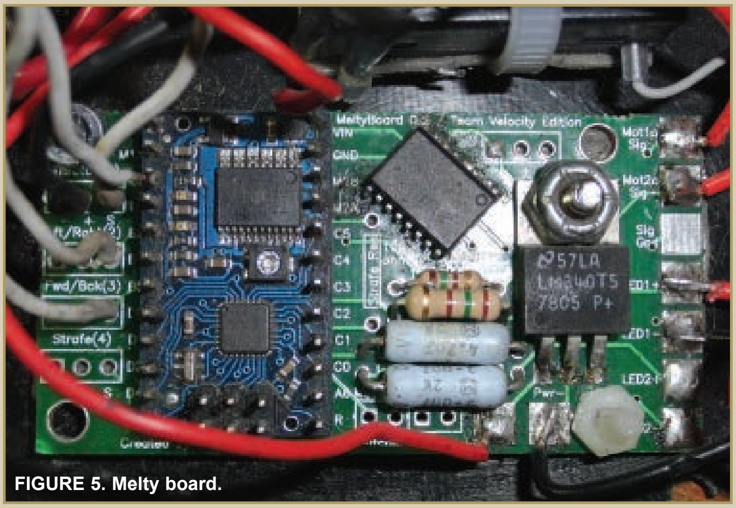


FIGURE 5. Melty board.

(a commercial version due to size). Motor control is via STMicroelectronics BU941ZT Darlington drivers, and the accelerometer is a Freescale 200G MMA2301EG.

Open Melt

The author, having rerun sufficient material to add pocket jingle without seriously enraging his editor, now begins to generate new material, at considerable more effort to himself ...

Rich provides information on his robots and other projects at **www.nothinglabs.com**, and has now upgraded and refined the software into a fully open platform. He says: "I shifted from Bascom AVR to C (WinAVR) to eliminate the dependency on commercial software. The full version of Bascom AVR is about $130 — which is a lot relative to the hardware for the project (which is only around $50-$100)."

The code is licensed through Creative Commons to acknowledge his authorship. Rich also has posted a schematic (**Figure 3**), a tutorial video, and well documented code at **http://nothinglabs.com/nothing/openmelt/**.

Also on the site is a link to a centrifuge calculator, links to parts sources, and a video comparing wheel types. He feels someone with moderate electronics/microcontroller skills could build a test platform — similar to that shown in **Figure 4** — in a weekend. The photo shows a test bed built by Rob Glidden, author of the companion article in this issue. He provides a nice segue into the next section, discussing how Rich's brain melt has spread.

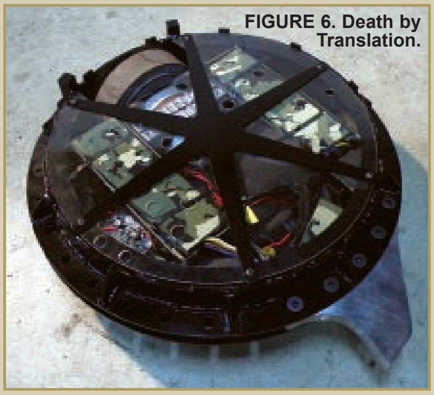


FIGURE 6. Death by Translation.

Open Melt is Contagious

Showing impressive literary skills, the author cleverly employs a "hook," neatly tying this section to the eye catching article title ...

Rob took Rich's ideas and went a step farther down the road. He used Rich's original Basic code as a reference; I've done all of my work using custom circuit boards and programmed in C. He says: "There are some neat tricks one can do to get more out of the Melty bot (control wise) but the biggest problem with putting a Melty bot in combat is making sure the insides stay where you put them — especially during hits!"

Kevin Barker collaborated with Rich to build a Featherweight (30 pound) combat bot. Kevin did the hardware; Rich did the electronics. They used an impressive S28-150 Magmotor, two HFS33 solid-state relays, an MMA2301, 200G accelerometer, and a Pololu Baby Orangutan B-328 microcontroller. The bot — "Death By Translation" — was a creative success but a combat casualty. Kevin says: "The bot actually did not work very well. It spun plenty fast but due to a couple factors, it didn't translate very well. It also was not strong enough to absorb its own impacts and bent several frame pieces in a huge hit with Touro Feather." **Figures 5** and **6** show the custom electronics board and the finished bot. Many more pictures are available at **www.teamvelocityrobotics.com**.

At least two other bots have recently been — or are being — developed using this technology. Stay tuned to your nightly news to see if the Melty virus becomes epidemic! **SV**

Information and photos for this article were provided by Rich Olson, Rob Glidden, and Kevin Barker. The Melty Brain logo was drawn by Sean Canfield.

Dual-Differential RPM Sensing or a Melty Brain/Translational Drift Robot

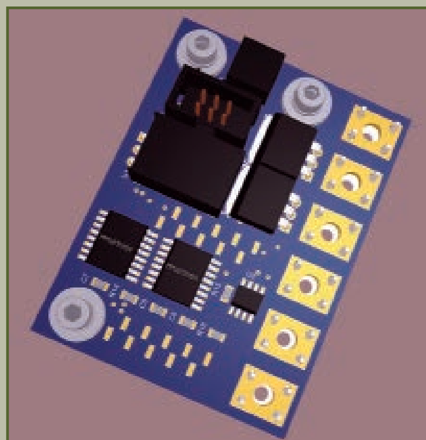
● by Rob Glidden

While there are many ways to tell your Melty bot's microcontroller how fast it's spinning, an on-board accelerometer is a particularly compact solution that can be built right onto the control board. The accelerometer measures the centrifugal force which is a function of the angular velocity (RPM) of the bot. From this, one can time each rotation precisely using the µC. My first Melty bot was based on Rich Olsen's single-accelerometer tracking mechanism, but I wasn't quite happy with the amount of calibration steps required. I've found an elegant way to address this issue that I think has some merit in a bot.

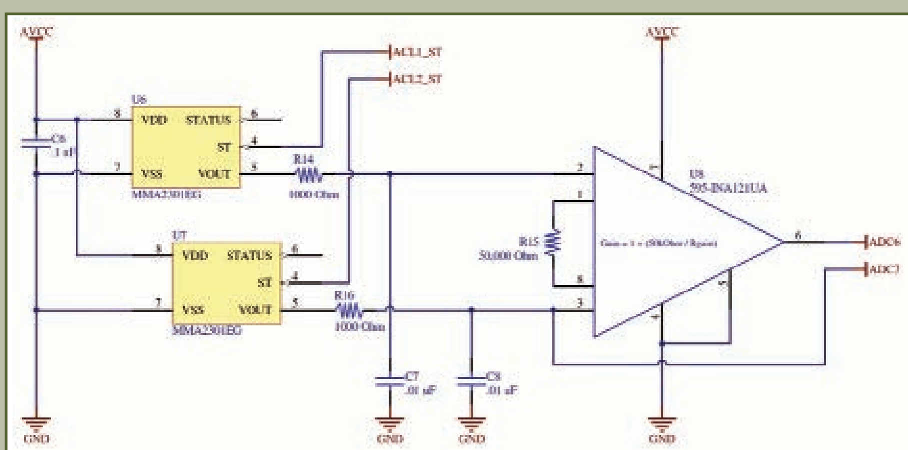
In single-accelerometer tracking, the RPM is given by RPM = $k \sqrt{\alpha_1/r_1}$, where k is a constant, α is the centrifugal acceleration, and r is the distance to the center of mass — theoretically it's also a constant, but in practice this is not true. You must know the distance to the Melty bot's center of mass (projected along the axis of the accelerometer). If you don't know it, your bot won't track properly. The actual value of r_1 is hard to predict even if you build your bot perfectly, and merely replacing the batteries or moving a couple wires can force you to re-estimate in smaller bots.

This is where dual-differential accelerometer tracking helps. Take a second accelerometer and place it a fixed distance behind the first one, preferably on the same circuit board, along a common axis. Because it is farther from the center of mass, it will therefore always read higher than the first one. Doing some math, it can be shown that an alternative formula for the RPM is given by RPM = $k \sqrt{(\alpha_2 - \alpha_1)/(r_2 - r_1)}$. This result is independent of the actual values of r_1 and r_2; the only important thing is the difference between them.

If you placed the two accelerometers as described in the image, you know this distance (r_2 - r_1) will never change. So, you can enter it in as a constant in the Melty bot's control program. You still have to subtract the acceleration values ($\alpha_2 - \alpha_1$) when the robot is spinning, however. The simple solution is to connect the two accelerometers



The physical hardware might look like this image. The two 16-pin SOICs in the lower left hand corner of the board are a pair of MMA2301EG accelerometers produced by Freescale; each rated for ±200 g. Their sensitive axes are aligned, and they are separated by a fixed distance of 0.50 inches. This distance is a constant regardless of where the board is located in the bot. Dual-differential control will actually work just fine with no code changes, even if the accelerometer's axes are not directly pointed towards the center of mass.



In this example, a Texas Instruments INA121UA instrumentation amplifier takes the difference in the signals and boosts it by a gain factor of 2. This gain can be varied with different resistors at R15, allowing you to tune the output signal to make use of the full range on your ADC. Here, ACL1_ST and ACL2_ST are the self-test pins for the two accelerometers. ADC6 is the main output and ADC7 is included just in case one wants to revert to single-accelerometer tracking. AVCC is +5 VDC. Each accelerometer's output is RC filtered to help eliminate noise.

with an instrumentation amplifier, using the **schematic** shown here.

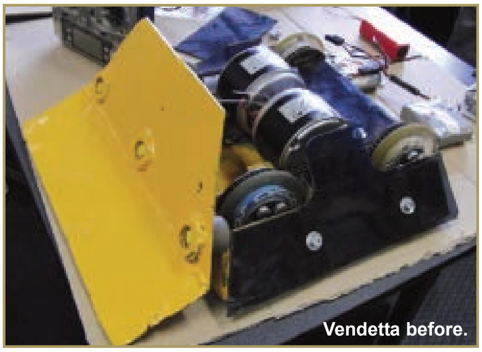
The net result is that the Melty bot controller no longer needs to be recalibrated with a new r_1 every time a part shifts slightly. In fact, this implementation never needs to know the actual distance to the center of mass. By removing this unknown as a factor in the control system, your bot will no longer lose tracking after the first hit takes a notch out of the weapon. It's a major step towards making a Melty bot robust enough for a combat application.

Note that if you tried to do this in software — first converting each accelerometer's voltage to an integer and subtracting to get the difference ($\alpha_2 - \alpha_1$) — you'd have very poor results. That approach would use less of the ADC's range than a single-accelerometer solution, and the conversion errors from α_1 and α_2 would add together, making the result even more inaccurate! **SV**

COMBAT ZONE'S GREATEST HITS

● by Kevin Berry

This month, only one builder submitted to Greatest Hits. Going by the mystery handle "kkeerroo," this Australian fighter highlights his bot — "Vendetta" — competing at RoboWars 7 in a semi-final fight against a spinner call Bender.


Vendetta before.


Vendetta after.

Get your greatest hit in print! Before and after photos, a brief description of the fight, and the builder's name can be submitted to me at LegendaryRobotics@gmail.com. Or, if you have an action shot that clearly shows what's going on, those are welcome too! These don't have to be current. You can (legally) submit anything, clear back to the good old days, when bots were made out of wood and builders were made out of iron. **SV**

EVENTS

Upcoming Events in June and July

Schiele Museum "Clash of the Bots" will be presented by Carolina Combat Robots in Gastonia, North Carolina on July 24, 2010. Go to **www.carolinacombat.com** if you would like further information.

Roaming Robots will present a show at Guildford on June 13, 2010. Go to **www.roamingrobots.co.uk** if you would like further information.

UBAYA RoboGames 2010 will be presented by the Universitas Surabaya in Surabaya, Indonesia, on July 31 to August 1, 2010. Go to **www.elektroubayarg.com** if you would like further information. **SV**

ELEKTRO UBAYA RoboGames 2010

EVENT REPORT: 2010 Central Illinois Bot Brawl

● by Dan Toborowski

On March 6, 2010 robots invaded Peoria, IL as the Central Illinois Robotics Club (CIRC) hosted their annual Bot Brawl event. This year's event was the club's largest yet with eight separate competitions, as well as Mech Warfare demonstrations. More than 20 robot hobbyists came to put their creations to the test in 1 lb and 3 lb combat, LEGO Sumo, mini 500 gram Sumo, 3kg Sumo, line following, line maze, and the Best in Show competition.

The CIRC has hosted all of these robot classes at past events, but 3 lb Beetleweights had not been offered since 2004 after being deemed too destructive for the 8' x 4' combat arena. In order to once again support these 3 lb heavyweights, as well as Mech Warfare demos for the 2010 event, the combat arena was upgraded to a new 8' x 8' battlefield with brand new inner barrier safety walls to provide more space and protection. It didn't take long, however, before the robots had left their mark on the brand new paint job and polycarbonate panels.

The day began with the Antweights competing in a double elimination tournament. Four of the five competitors featured powerful saw blades, and the fifth robot used a servo-powered clamping arm to grab its opponents. Some of the robots were upgraded versions of previous competitors, while others were competing for the very first time. For a few, the day of the event was the first time they were even fully functional. Each robot was able to put up a decent fight but not everyone left in one piece.

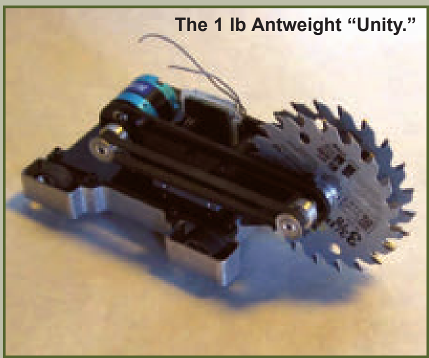
Upgraded 8' x 8' combat arena.

The saw blade robots put on a great show and did a significant amount of damage, but in the end it was "Death Grip III" with his clamping arm that took first place. Piloted by Matt Julien, this robot quickly made short work of every competitor thanks to its robust design, quickness, excellent traction, and effective weapon. James Frye's "Unity" came in second place with its powerful saw blade and reliable Inertia Labs chassis. Perhaps Unity's best showing was during the rumble match in which he single-handedly disabled every other robot with relative ease.

Another sawblade-equipped robot named "Buzzsaw" took third place despite being finished by builder Curt Boirum around 3:00 AM that morning. All of the competitors put up a good fight, but the inability to self-right or a less powerful drivetrain eliminated everyone except for Death Grip III.

The 3 lb combat competition had a smaller turnout than the 1 lb Antweights but still provided plenty of excitement. A total of three robots entered, including two multi-bots and a remote controlled toy. One of the multi-bots was composed of two identical 1-1/2 lb wedges, while the other multi-bot was made up of three Antweights who individually placed first, second, and third in the 1 lb tournament. Most of the 3 lb Beetleweight matches looked more like a rumble because there would be up to five robots zipping around the box with parts and sparks flying.


The 1 lb Antweight Champion "Death Grip III."


The 1 lb Antweight "Unity."

The 3 lb Beetleweight Champion "Pro-Unic-grip."

The 3 lb Beetleweights "Thing 1 & Thing 2."

The audience loved watching the robots work together, as well as the occasional accidental friendly-inflicted damage. The three-part multi-bot "Pro-Unic-Grip" ended up victorious and took home first place. The other multi-bot "Thing 1 & Thing 2" won second place, and the RC toy "Why Not" took third place.

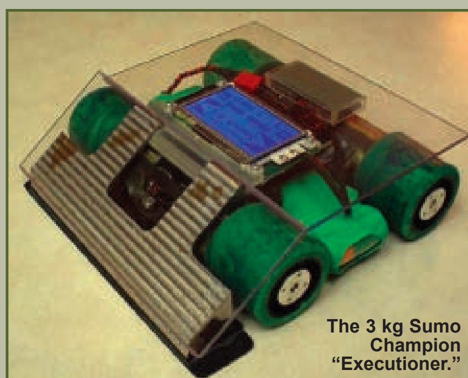
As always, the combat robots were one of the most popular competitions but the spectators also really enjoyed the special Mech Warfare demos. Andrew Alter, the creator of the Mech Warfare competition, offered to come to the Central Illinois Bot Brawl and showcase these exciting robots. A few other builders brought their robots including James Frye from Lynxmotion and Mike Ferguson who won the inaugural 2009 Mech Warfare competition at RoboGames with his robot Issy.

These walking robots face off against each other with airsoft guns and wireless cameras, and use sensors to detect how often they've been hit by their opponent. The rules specifically require that operators use only the video from the on-board camera to direct their robots rather than viewing the match directly. Because of this, competitors are able to strategically maneuver around each other and flank their opponent before they even know they're being targeted.

The two biped robots were not ready to compete in these demonstration matches but the other two quadruped robots put on a terrific show. By the end of each match, the combat arena was filled with colorful plastic airsoft pellets as a testament to the intense preceding battles. Judging by the audience's excitement, it's fair to assume that Mech Warfare will return at future Bot Brawl competitions.

For the complete results, and more pictures and video from the event, visit **www.circ.mtco.com**. **SV**

The 500 g Mini Sumo Champion "Exspurt."



The 3 kg Sumo Champion "Executioner."

The CIRC would like to thank everyone who helped make this event possible including the many generous suppliers who donated prizes. These include Basic Micro, Fingertech Robotics, LabJack, Lynxmotion, MaxBotics, Ortech Education Systems, PNI Corp, Pololu, PROBOTIX, SchmartBoard, *SERVO Magazine*, Sherline, Smart Robots, Solarbotics, Technological Arts, Vantec, Wright Hobbies Robotics, and Zagros Robotics.

Give Your Robot A Better Grip On Things

By Anthony Cunningham

Humanoids have fascinated me for many years, and I suppose that would also include the majority of the world population, as well. Honda's Asimo got my attention, but the Robo-One competitions really got me hooked. Watching them fight each other on the Internet only escalated my passion. My interest in humanoids has grown into an obsession ever since.

I'm known for being quite the robot fanatic, to the point it drives everybody I know crazy. Let's face it, the robot community is young and has everywhere to grow, and since robotics is all about efficiency, knowledge and experience must be passed on to others — which only makes this hobby flourish. Therefore, I am here to impart my knowledge and passion upon you, so hopefully this article will help spark your imagination.

Hands Versus Grippers

I've made a few grippers for my robots, and they have been very effective. Although grippers are fairly easy to make — and they have their place in robotics as a whole — they're just not as cool as digits. Grippers have no real personality, and they're kind of plain. I mean no offense to those of you who have grippers on your bots. Like I said, I've made many a gripper, but I want to step it up a notch.

Hands have got to be one of the hardest features to duplicate. With so many joints and tiny parts, anybody in their right mind would quickly give up, especially if you don't have a visual of a good hand design. Also, being able to implement a power source to cause the fingers to grip and keep the bulk of your design to a minimum is a challenge. Our hobby deals with small parts, but we're not clock makers! Once you master the art of hand making, your robot hobby will soar to a whole new level. In fighting competitions, nothing says "awesome robot" like having your robot ball its fist for that knock-out punch! Plus, hands help give a humanoid more personality. Let's get started!

FIGURE 1.


FIGURE 2.

Gathering Our Materials

The perfect place to gather up the materials for this project, is the local hardware store. Trust me, this type of store is definitely the place to go to find inspiration for new ideas and/or to improve existing ones. That's how I came up with this design for my hands. Everything I needed was right there in the store.

Start in the nuts and bolts section, even plastic nuts and bolts. My store had plastic washers that were 3/8" thick, 1-1/4" in diameter, with a 3/8" hole in the middle (**Figure 1**). I thought this hole was going to be a problem, but it actually worked out perfect. (More on that in a bit.) This plastic washer served as the palm of the hand.

Next, I needed something for the digits. Something flexible, but something that could hold its own and return straight. I found some clear, flexible 3/16" OD tubing on a spool that was sold by the foot. I got three feet of it, which was way more than I needed, but at a dollar a foot I couldn't go wrong. So, now I had the fingers.

My next concern was what to use to pull the fingers with. I must have walked around that store for 30 minutes, trying to figure out what to use. A store employee asked if I needed any help, so I told him, "I need something to pull the fingers in ... something lightweight but strong." I didn't figure he could help me (and was going to dismiss any answer he came up with) but then he says, "what about fishing line?" You could have knocked me over with a feather. Why didn't I think of that! Needless to say, I thanked him *a lot* and picked up a spool of 6 lb fishing line (it was the only strength they had) for just a couple bucks. You could use a higher rated monofilament if you want. (You could even cut some off of your fishing pole.) Lesson learned: Never, ever underestimate someone else's ideas, no matter who they are.

The little black items you see in **Figure 1**, are little plastic sleeves that go on the ends of bolts — another suggestion my newfound hardware store friend came up with, now that he knew what I was doing. These will serve as the finger tips.

Most hardware stores will carry products from K&S Metal for hobbyists. One sheet of .032" thick aluminum will work just fine. They come in 6" x 12" sheets. I got my aluminum from a scrap metal yard. You'd be surprised what you can find at these places. Like, they say, "one man's

junk, is another man's ... robot."

You're also going to need a servo for the hand. The servo doesn't have to be a powerhouse; it just needs at least 45 oz/in of torque, which is what I'm using. Currently, it is the TS-53 standard servo from Tower Hobbies.

Besides being a robot fanatic, I'm also hot glue junkie. I use it for everything ... well, almost everything. I live in the northern part of the US where the temperatures are not that hot all the time, so using hot glue for keeping things together is not an issue. For those of you who live in warmer regions, you might want to use either a high temperature hot glue stick or go with epoxy. When it gets too hot — say around 90 degrees — the glue will get soft and eventually cause your creation to fall apart. I'll show you a little technique I do to keep things together a little longer, in a moment. A nut and bolt alternative is more durable, but using this method can add extra weight to your humanoid.

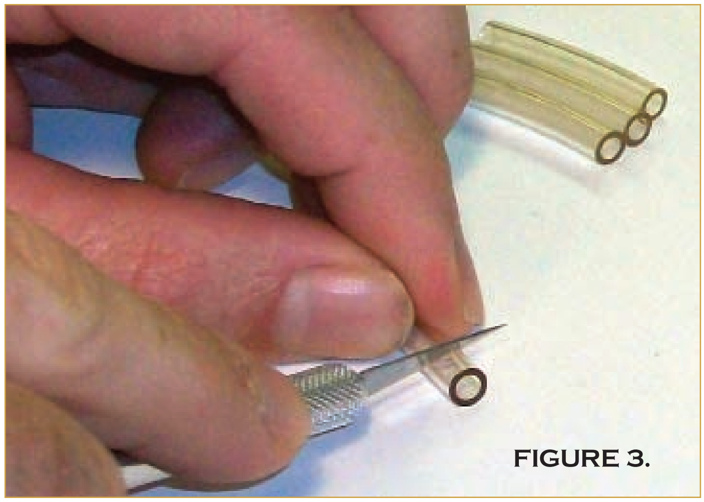


FIGURE 3.

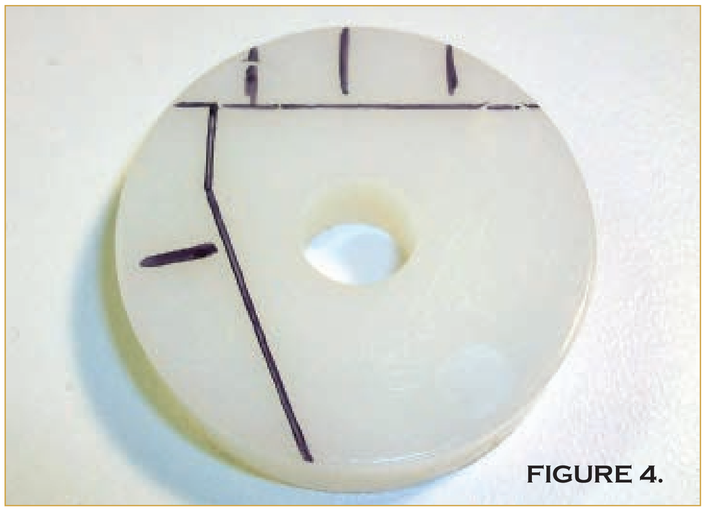


FIGURE 4.

Finger Food

I call this section finger food because this is where your cutting skills will be used. *Extreme care must be taken here.* Let's begin work on one hand. In **Figure 2**, I'm cutting the tubing into 30 mm lengths. I usually work in millimeters for simplicity and accuracy's sake. Three fingers and a thumb should be enough for grabbing things, so you need to cut four pieces. If you decide to make the fingers longer or go with an extra finger, it will be harder for the servo to pull the extra resistance — so get a stronger servo to do this. I've found that too long of a finger looks creepy and adding an extra finger is really unnecessary.

Next, we need to cut "v" notches in the tubing. Like human fingers, there are three joints per finger which means we need to pain-stakingly cut three notches into each tube finger. *Please be careful!* Take your time with this part. Make sure that you cut the v notch pretty wide, in fact, the wider the better. A wider notch makes it easier for the finger to bend. Also, make sure that your cut goes all the way to the bottom of the tube.

You're probably wondering why we even need the notches. The notches give the finger a "direction" to move. If you do not cut these, your finger will just bend willy-nilly in any direction. There is also much more resistance to make the finger bend. Another thing to keep in mind is to make sure your notches are in line with one another. Otherwise, your finger is going to bend in different directions (**Figure 3**). It's easy to misalign the cuts when you're working with something so small, so take your time.

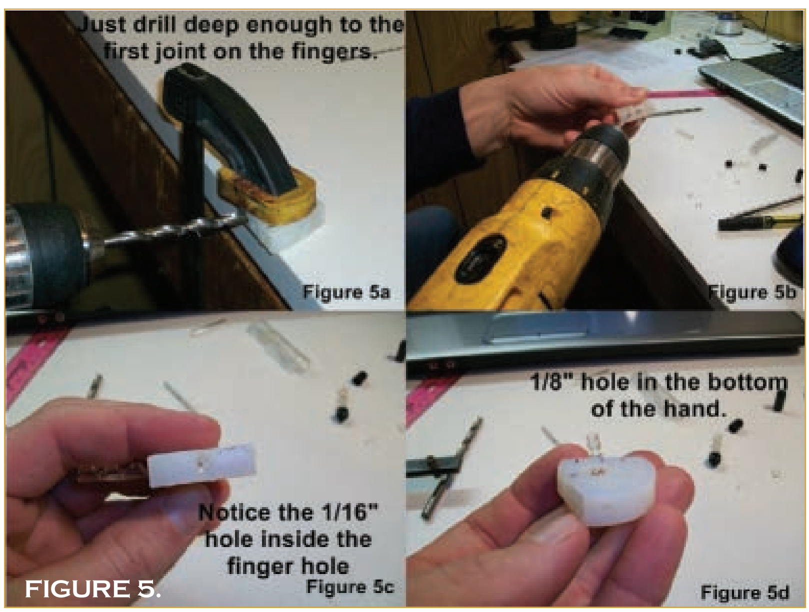


FIGURE 5.

Next, we'll cut out the hand and drill holes into it for the fingers. **Figure 4** shows how I've marked on the plastic washer where I want to make my cuts. This plastic is very hard, and cutting is very difficult. I have a band saw that I use to cut small things like this. A scroll saw will also work. The plastic tends to melt and harden quickly, so use a slower speed to cut it. If you don't have these particular tools and don't have friends or neighbors who have them either, you could use a sander or a hand file. Or, you could just leave the plastic the way it is (giving it a cute, little round hand). Now we need to drill the holes into the plastic to put the fingers into. Make sure that the drill bit you are using is the diameter of the tubing (or a little smaller), so you get a snug fit (**Figure 5a**). It's a good idea not to glue it in permanently. I'll explain why in a bit. Drill only the depth of the first joint of the finger. Look at your hand; the first joint is at the palm of your hand. Test the depth of the finger as you drill (unless you have one of those fancy depth stops for your drill bit).

FIGURE 6.

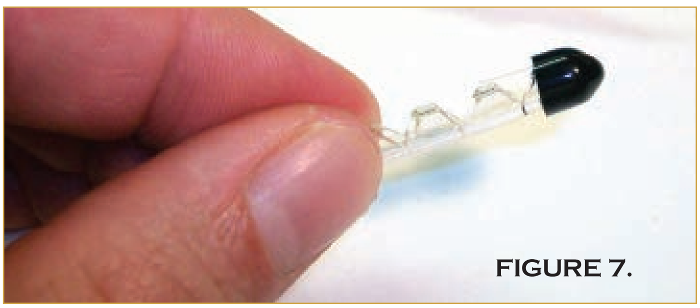
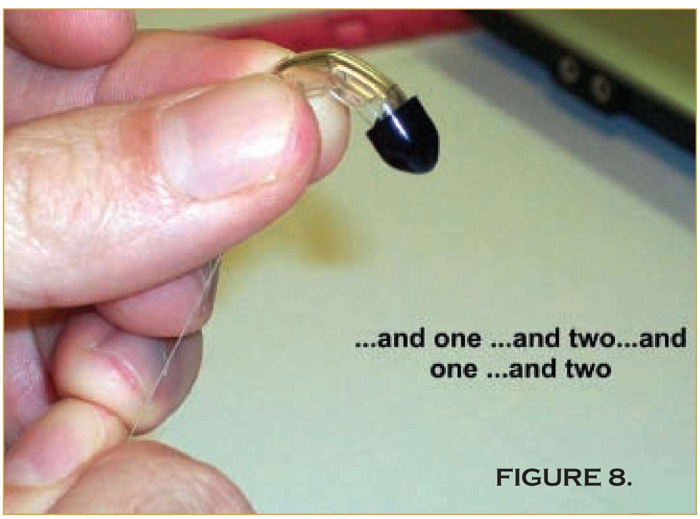


FIGURE 7.



...and one ...and two...and one ...and two

FIGURE 8.

Once you get the holes for the fingers drilled, you need to drill a 1/16" hole towards the middle hole, through the finger holes you just made (see **Figure 5b** and **Figure 5c**). Next, drill a 1/8" hole in the bottom of the hand towards the middle hole (**Figure 5d**). All the fishing lines will come out of the 1/8" hole. The middle hole (that I talked about earlier) ended up being perfect for "fishing" the line in through the hand. If you use something other than the plastic washer I used, you might consider drilling a hole in the middle so you can see what you're doing.

Putting It All Together

Our next step is to take the fishing line and cut about a foot for each finger. Then, we take a pair of needlenose pliers and bend an inch of it on one end and crimp it, so that it keeps a hook-like shape. Feed the fishing line in through all of the fingers, letting the hook of the fishing line rest on the tip of the finger (**Figure 6**).

If you couldn't find the bolt end covers, you can hot glue the inside of the tip of the finger. This should hold the fishing line in place. I know that hot glue gets a little messy, but you can trim any excess after it cools. If you do have the bolt end covers, then cut them so they are about 10 mm long from the tip. Put a drop of super glue (**Figure 7**) on the tip of the robot finger (not your finger!) and the hook of the fishing line, and then slip on the rubber tip. Make sure the hook of the fishing line is on the bending side of the finger as shown.

After the glue sets, you should be able to hold the finger in one hand, pull the fishing line with the other, and make the finger bend (**Figure 8**). Now, feed the fishing line all the way in through the hand. Put the fingers into the holes you drilled for them; they should fit snuggly. I did not glue the fingers into the hand because if the fishing line should break, it's going to be difficult to feed a new line into both the finger and the hand at the same time.

Figure 9 shows a completed hand. You must lay out the aluminum with the hand and the servo where you want them. Trace the hand and the servo so you can cut the excess aluminum. Use a pair of tin snips. Gluing the hand to the aluminum didn't seem like a strong enough idea, so I screwed the hand to the aluminum. Make sure you don't drill into your fishing line, however! On the other hand (pun intended), I attached the servo with hot glue. If you would rather mount the servo to the aluminum, then you will

need to make tabs on the metal. Simply set the servo on the aluminum and mark where the servo mounts should be (**Figure 10**). Cut the aluminum along your lines to make the tabs; mark and drill the holes. I drill holes about 1/16" smaller than the bolts. The bolt should tap (making the threads into the hole) right into the hole, so you don't have to use a nut and washer (**Figure 11**).

If you're hot gluing, you need to make score lines with a knife on the side of the servo and on the aluminum where the servo is to go. Heat the side of the servo and the aluminum with a heat gun so the hot glue sticks to the score lines. Scoring lines into the plastic and the aluminum gives the glue something to hang onto and enhances its gripping ability. If you don't do this, things will easily come apart. You need to heat the aluminum, otherwise the hot glue will harden too fast because the aluminum disperses the heat of the glue so well. As you can see, I also glued the forearm servo to the aluminum.

For the fishing line, I used a round servo horn and added a thick, round plastic washer to it for pulling the fingers. I used a round surface for pulling because it gives the servo better leverage and can hold a tight grip with little effort. If you use a lever horn for pulling, the servo will have to strain to hold a grip position — draining precious battery power. Important note: Make sure you center your servo or position the servo where you want it before attaching the fishing line or you might break them. I already made that mistake once!

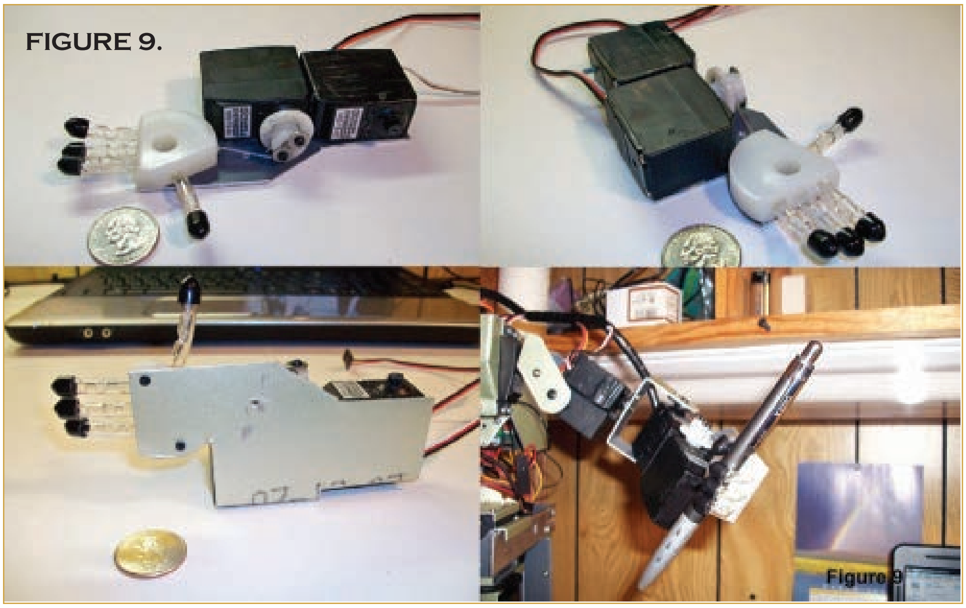


FIGURE 9.

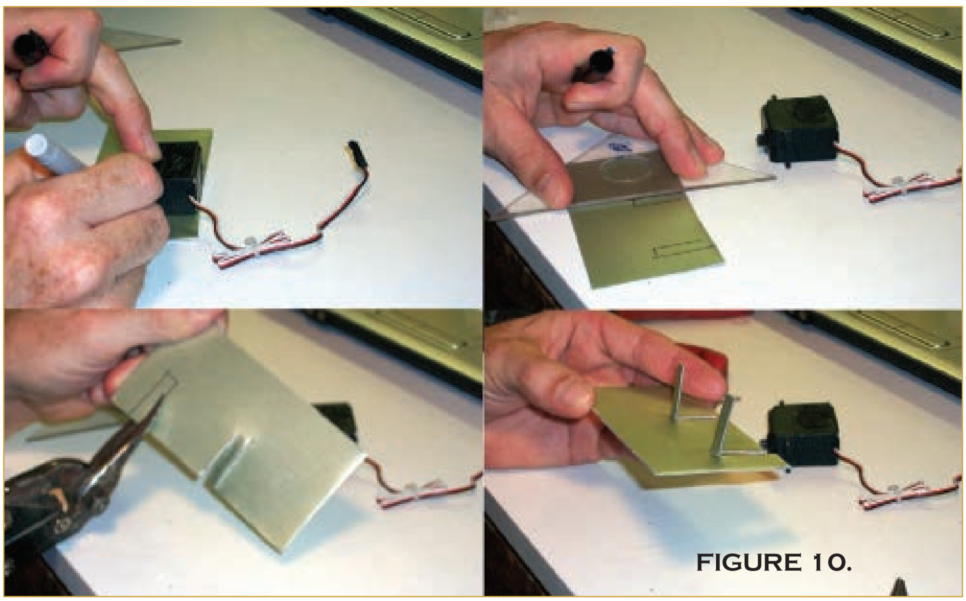


FIGURE 10.

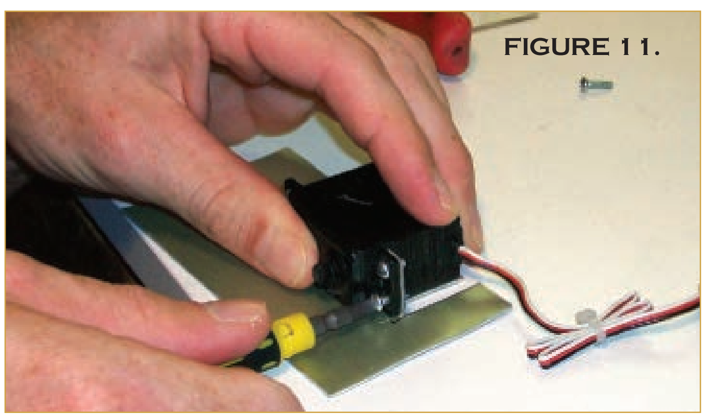


FIGURE 11.

Conclusion

Hands add a whole new dimension to your robot, giving it more of a personality. Consider making larger, full size hands by adding a servo per finger and making a potentiometer sensor glove so when you move your fingers, it responds accordingly. You hold the creative possibilities in the palm of your ... well, you know. **SV**

Enhance Your Plain Vanilla

Benchtop Power Supply With A PICAXE

There are several essential tools on any robotics experimenter bench. From an electronics standpoint, there are four: a programmer for the processor of your choice; a soldering iron; a multimeter; and a benchtop power supply. The latter is often substituted with either batteries and/or wall-wart adaptors. However, this is both an inflexible and unreliable arrangement, and eventually one finds a need for a "real" power supply. To do any serious design and development work, you need a supply with more features than an on-off switch. Otherwise, how do you test changes in performance with voltage fluctuations? Reliable power is required for work on vision systems, navigation, IR and US sensors, a wireless communications link, and more, and each system typically presents different power requirements. To get what you need, you can roll your own, purchase a full fledged and expensive lab grade unit, or build one of the many kits found on the Web. Usually, the end results will be a no-frills unit, with either fixed or variable voltage, and from one to three outputs. Voltage and/or current metering is a big plus, but rapidly becomes quite expensive. This article presents a cost-effective alternative.

This project is not exclusive to the Elenco kit; it can be used on any other plain vanilla power supplies, as long as the following criteria are met:

- *Common ground for all the outputs.*
- *An unregulated DC source around nine to 12 volts. This usually will be the unregulated source that feeds the +5V fixed regulator.*
- *A maximum of 1.5 amps on the variable outputs; three amps on the fixed one.*

If your supply only has two outputs, you'll only leave the unused monitoring channel unconnected.

By Fernando Garcia

I originally purchased an Elenco Precision power supply kit shown in **Figure 1**. This is a triple output job built around the ubiquitous LM317/LM337 for the variable voltages and a PNP-enhanced 7805 for the fixed one. It all comes in a sturdy and attractive steel case with all the necessary hardware, and can be purchased for around $60 US. This is one simple kit, reminiscent of the old EICO kits — functional but otherwise bare bones — that just begs to be upgraded. With an upgrade in mind, I designed a board based on a PICAXE 14M and a few other components which will add the following features:

FIGURE 1.

- Simultaneous voltage reading of the two variable voltage outputs and current reading of the fixed voltage output via an LCD screen. You don't need to tie up your multimeter every time you need to make an adjustment.
- At power-up, the supply starts with all outputs disabled. When enabling them, they are all enabled simultaneously — to prevent latch-up in multi-voltage applications.
- Audible alert when enabling/disabling the outputs.
- While the outputs are disabled, they still remain monitored to allow proper voltage adjustment before enabling them. This avoids accidentally frying the circuit with an incorrect voltage.
- Overload current warning via visual and audible alerts.
- Heatsink temperature monitoring which turns on a fan when a preset temperature is exceeded.

Circuit Description

Refer to the schematic in **Figure 2**.

Let's start with the power supply for this circuit. A nine to 12 volt DC unregulated source is fed via connector J4 to adjustable voltage regulator U4. Related components P1, R2, and R3 set the voltage to precisely 5.12 volts. The reason for this oddball voltage instead of a nice round 5.0 volts is that the PICAXE utilizes its supply voltage as its reference. With its 10-bit internal ADC (analog-to-digital converter), this reference voltage will provide a resolution of exactly five millivolts per step that greatly simplifies calculations. R1 and D1 provide a voltage of roughly +6 volts which is used to allow enough input common mode range for the current-measuring op-amp. The +5.12 volts is also used for most other devices, but it is also converted to a negative voltage via U3, C3, and C4. This negative voltage is required by one op-amp section which will monitor the negative voltage. R0 isolates the power and sense grounds, but allows a large degree of continuity if one becomes disconnected.

Circuit Operation

Since the PICAXE is the heart of the circuit, it is best to describe the circuit's operation from its individual pin functions. Let's start with the input pins.

Input 0 connects to an external pushbutton switch. When pressed, a logic high on this pin will cause the software to toggle the output relay, emit a brief tone by the piezo buzzer, then update the display accordingly.

Inputs 1, 2, and 3 are not used and thus grounded.

Input 4 is the ADC input and will receive the four input variables sequentially as selected by the analog switch U2 as described below. This selection is necessary as the PICAXE 14M only has two ADC channels.

The analog variables to be read are: variable positive output volts, variable negative output volts, fixed output current, and heatsink temperature.

Output 0 is the serial output pin that connects to a serial enabled 2x16 LCD display. I used a display from Scott Edwards Electronics which provides a nice bezel and mounting kit, but you can substitute it as long as it supports the "254" command set and N2400 baud.

Outputs 1 and 2 are used to select the four analog signals via an analog multiplexer U2. On the multiplexer, four channels out of eight are employed. The temperature signal comes from U6, an LM34 temperature sensor that attaches to the heatsink and connects to the main board via J3 and is filtered by R20, C9. The positive output voltage comes from a voltage divider formed by R4 through R7. The negative output voltage requires an op-amp U5a and resistors R8 through R12 which provide a gain of -1/4 to invert and attenuate it.

Finally, the fixed output current is sampled with a low value resistor R17 and amplified via U5b and its gain setting resistors R13 through R16. Since the resistor samples the voltage from the positive fixed output — which could be as high as 5.25 volts — we require powering the op-amp with a voltage slightly higher than that. The op-amp is a rail-to-

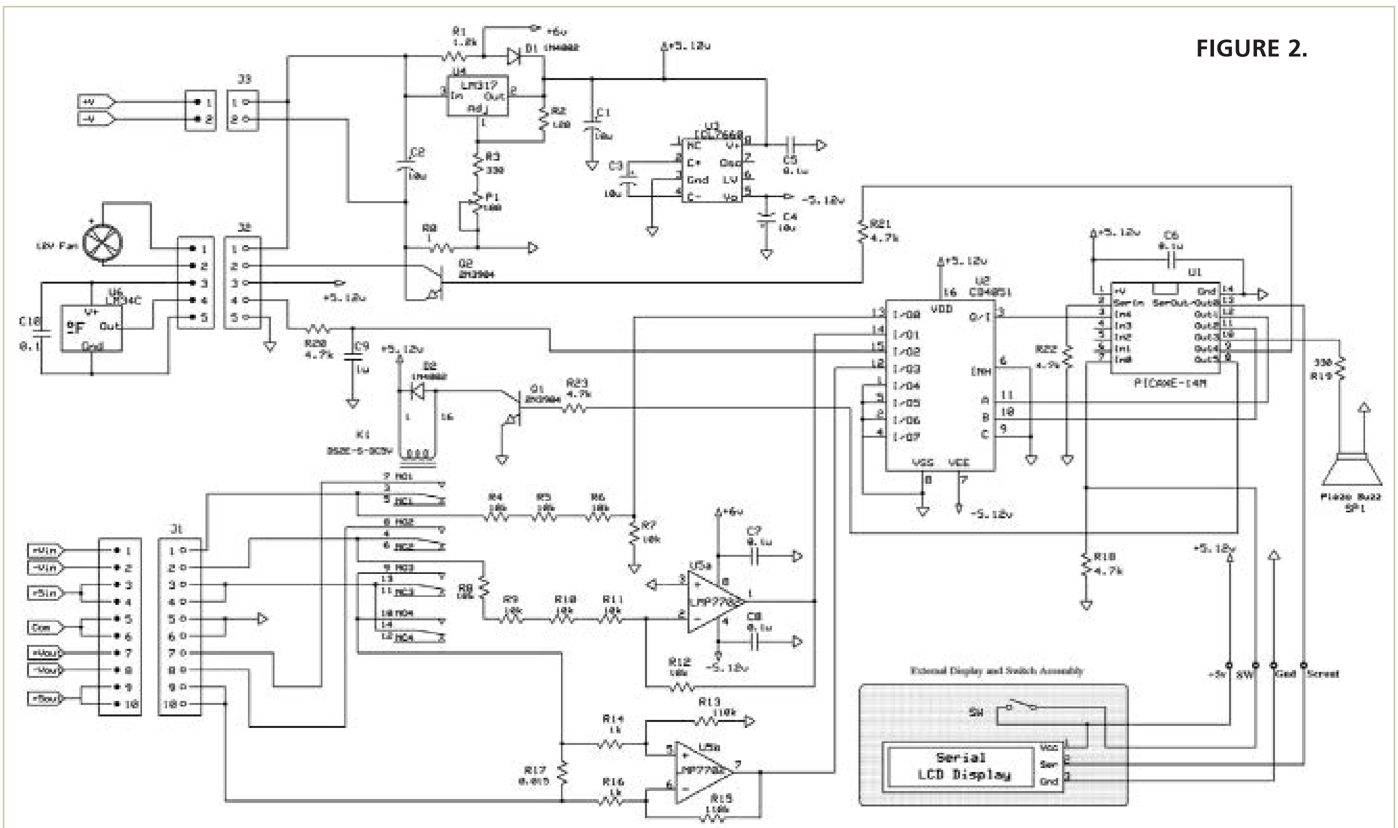


FIGURE 2.

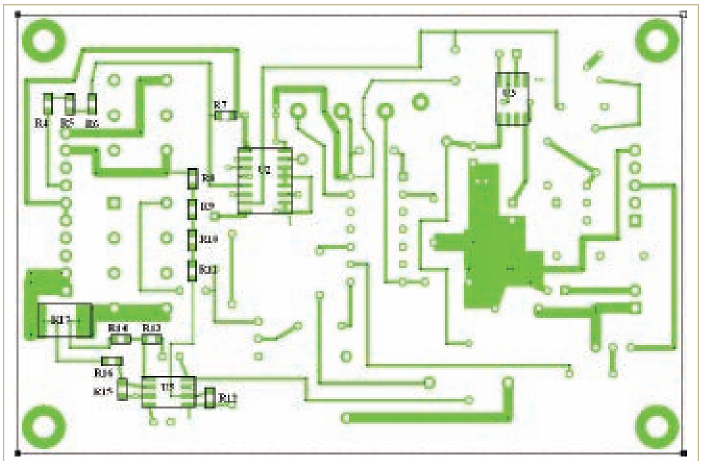


FIGURE 3.

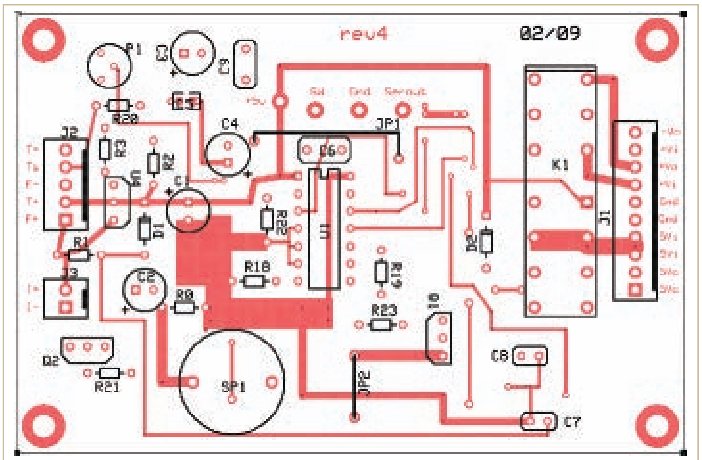


FIGURE 4.

rail device, but cannot amplify voltages sitting higher than its positive supply (thus, the need for the +6 volts).

Output 3 is the piezo buzzer output. Resistor R19 is only used to adjust its volume, and you may adjust its value. Do not eliminate it completely, although I have never had any problems driving a piezo directly. Its large capacitance could introduce some noise back into the PICAXE.

Output 4 drives — via transistor Q2 and connector J3 — a fan which will be enabled once the heatsink exceeds a certain temperature. The fan must be a 12 volt device with a current consumption not exceeding 350 mA.

Output 5 drives — via transistor Q1 — the output relay K1. This is a four pole, sensitive coil device. Its contacts are in series with each output and it can handle two amps continuously. Therefore, for the fixed output which can provide current in excess of three amps, two poles are paired in parallel to increase the current handling capabilities. Diode D2 protects the transistor. When the unit is first powered up, this output defaults to a logic low, making sure the outputs are disconnected.

You may be wondering why I've employed a four 10K resistor string (R8 through R11) and a three resistor string (R4 through R6) instead of a single 40K and 30K, respectively. The reason is that these voltage dividers set the project's accuracy, and thus 0.5% tolerance devices are recommended. Low cost versions of these tight tolerance resistors are almost only available as SMT devices, which typically are sold in multiples of 10. Thus, it makes more sense to purchase a single value 10 resistor pack and arrange the resistors for the required value, rather than purchase several 10-packs of which most devices will not be used. A little additional soldering is all that is required.

Building the Circuit

This project consists of three assemblies. First, there's a power supply to be upgraded, whether you use a kit or roll your own. Second, there's the main project board where the microprocessor and signal conditioning resides. Third

Parts List

ITEM	DESCRIPTION
Resistors; 5% unless otherwise noted.	
R0	1 ohm, 1/4W
R1	1.2 K ohm, 1/4W
R2	120 ohm, 1/4W
R3	330 ohm, 1/4W
R4 thru R12	10 K ohm, 0805, SMT, 0.5%
R13, R15	110 K ohm, 0805, SMT, 0.5%
R14, R16	1 K ohm, 0805, SMT, 0.5%
R17	0.015 ohm, 1220, SMT, 1%
R18, R21, R20, R22, R23	4.7 K ohm, 1/4W
R19	330 ohm, 1/4W
P1	100 ohm potentiometer
Capacitors:	
C1, thru C4	10 µF/16V electrolytic
C5	0.1 µF /50V ceramic, 0805, SMT
C6,C7, C8, C10	0.1 µF /50V ceramic
C9	1 µF/ 50V ceramic
Semiconductors:	
D1, D2	1N4002, general-purpose diode
Q1, Q2	2N3904, NPN transistor
U1	PICAXE-14M controller, SOIC
U2	CD4051, analog switch, SOIC
U3	ICL7660, voltage converter, SOIC
U4	LM317L, adjustable regulator, TO92
U5	LMP7702, rail-to-rail op-amp, SOIC
U6	LM34C, Fahrenheit temp sensor, TO92
Miscellaneous:	
K1	4P2T sensitive coil, 5V relay, Panasonic DS2E-S-DC5V
SP1	Piezo buzzer, external drive
SW1	N.O. pushbutton switch
J1	10 position header, 0.100" centers
J2	Five position header, 0.100" centers
J3	Two position header, 0.100" centers
F1	12 volt fan (*see text*)
LCD1	Serial 2X16 LCD display, 2400 baud (Scott Edwards Electronics BPK-216 or similar, *see text*); **www.seetron.com**
Case	PacTec PS24-150 or similar
PICAXE Software	Powersupply_4D.bas (available at **www.servomagazine.com**)

are the peripheral accessories that are not mounted on the main project board. These include the serial LCD display, the switch, the temperature sensor, and fan(s). Let's describe the LCD display and switch assembly.

Of course, one of the main considerations of any project is its cost. The most expensive item is the LCD display, so it pays to shop around. You'll find them anywhere from $20 to $60. There are many 2x16 serial LCD assemblies on the market. As long as it supports the standard 254, one command set, and receives at 2400 bps TTL serial data, you are in business.

The selection (at least for me) mostly boiled down to a display that would provide a nice, professional looking assembly. I went with for a Scott Edwards BPI-216 which provides an optional mounting kit with faceplate and mounting hardware (BEZ-216) for a nice looking finish. This is important since both the switch and LCD mount on an external plastic enclosure, and while cutting out the panel to make the display window, you'll most likely leave ragged edges.

After you have mounted the LCD and switch, wire them with color-coded wires. Note that the LCD and switch share the +5.12 volt line.

Let's go over the main board. This board has both thru hole and SMT components that are mounted on both sides of the board. Exotic or ultraminiature packages were avoided; the chips are 0805 sized so they're large, and the ICs are SOIC which are the easiest to handle. Assemble the SMT components first, which all go on the bottom layer (**Figure 3**) with the exception of C5.

Make sure that you check the orientation of U2, U3, and U5; their pin 1 is clearly indicated on the board.

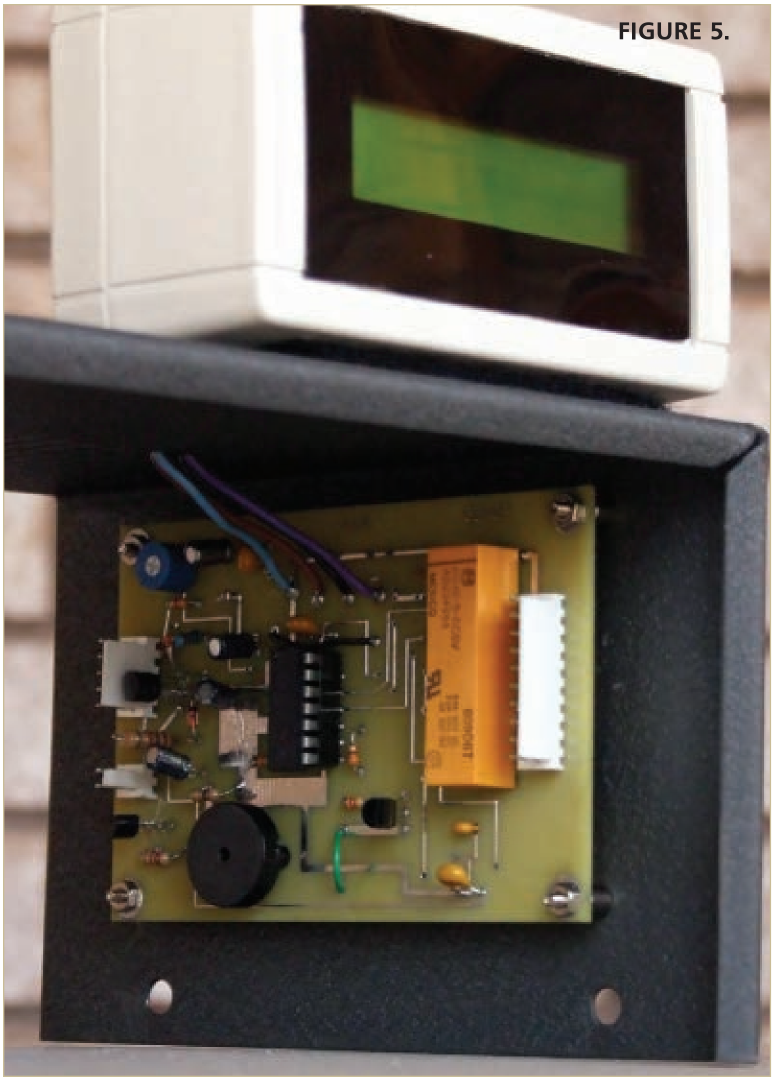


FIGURE 5.

Checking the Circuit

Once you finish the previous step and have thoroughly inspected everything for poor solder joints and bridges, proceed with assembling the thru hole devices on the top layer (**Figure 4**). Watch out for the proper orientation of microcontroller U1, the electrolytic capacitors and the diodes — all which have their polarity indicated with a square pad — and also that of relay K1. The relay's body is marked with a black line which must match the silkscreen shown in the artwork. Also careful with the connectors; they are keyed.

Once you have completed the assembly and thoroughly inspected it for soldering errors, apply nine to 12 volts at connector J4. Then with a digital multimeter, adjust potentiometer P1 for a regulator output of exactly 5.12 volts. The precision of this adjustment will determine (to a large degree) the voltage readout precision, so perform it carefully. Once you have done that, check the voltage on D1's anode. It should be close to six volts. Lastly, check the voltage on the negative pin of C4 which will measure close to -5 volts.

After the different voltages have been verified, temporarily solder the four wires coming from the LCD/switch box to the main board — to the +5V, Gnd, Serout, and SW pads. Connect the fan and temperature sensors — which (at the moment) should not be attached to the heatsink — by plugging them into connector J3. Apply voltage again to J4; this time, the display should turn on after a brief delay and show the "disabled" screen and 0.00 volts on both outputs. You may want to adjust the LCD display's contrast setting (if it's available). Push the button. After a brief beep and relay clicking, the display will change, showing the output current which should be 0.0 amps. The fan should still not be running. Put your hot soldering iron close to — but not touching — the temp sensor. After a few seconds, the fan will start to run.

Push the button again. After another brief beep and the relay clicking open, the display will revert to the disabled screen. If all the steps have been completed successfully, your project is essentially functional. Disconnect all connectors and desolder the display wires.

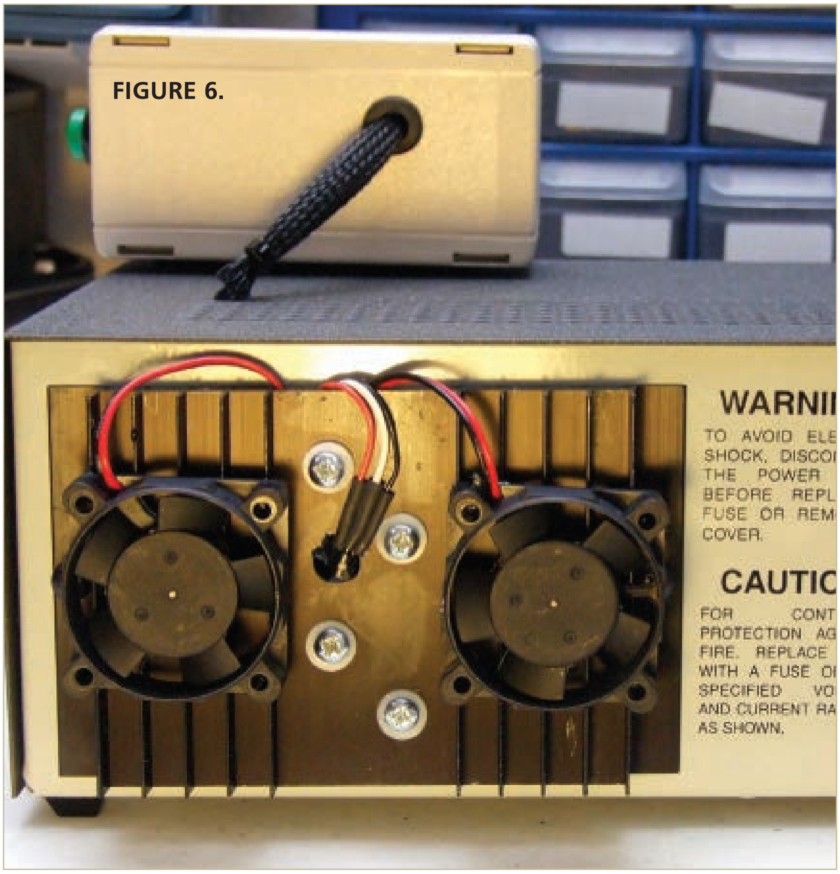


FIGURE 6.

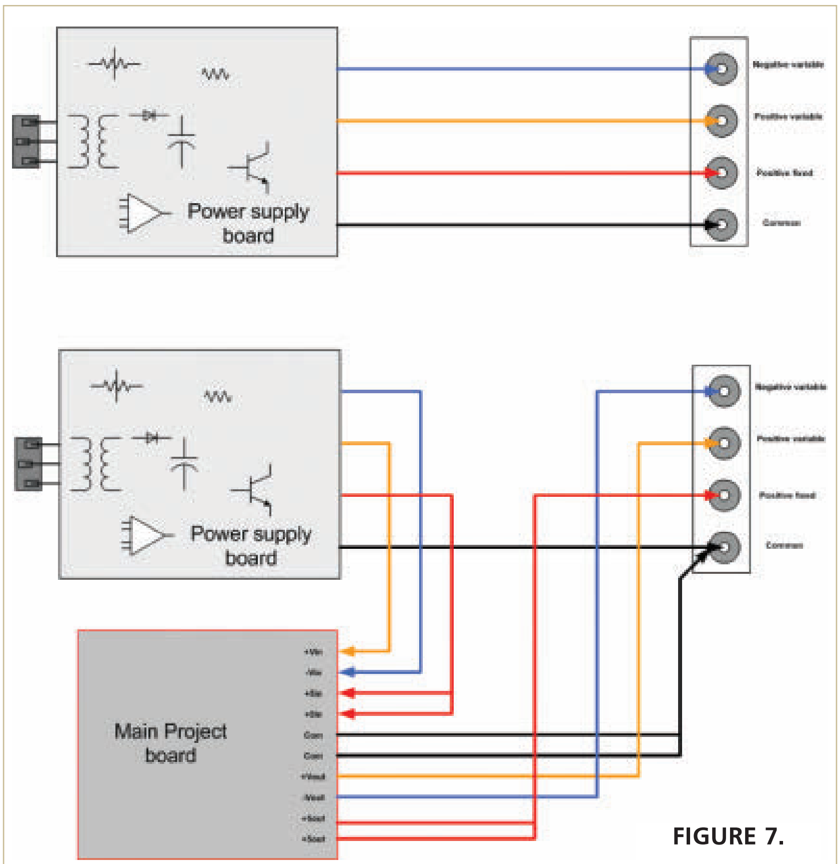


FIGURE 7.

Getting Attached

The last step is to attach the project into your existing power supply project. Drill four holes to the power supply cover to attach the main board with 4-40 hardware. Now permanently solder the wires that come from the display to the board. The display assembly is best attached to the main cover with Velcro™ tape which allows repositioning for the best viewing angle. After you have finished, it should look something like **Figure 5**.

Locate the power supply board on the main filter capacitor for the fixed output. It will be several thousand µFs and rated 12 to 16 volts. On its terminals, verify that you have a raw DC voltage in the 8.5 to 12 volt range. Solder the cables to the plug that connects to J4, observing polarity.

Attach the fan(s) and temperature sensor to the heatsink as shown in **Figure 6**. A two-part epoxy works well for this purpose. Just make sure that you don't glue the fan's blades shut! Depending on the actual heatsink, you may need one or two fans. In this particular heatsink configuration, two small fans were better than a single large one. They are wired in parallel. Just remember to observe the 350 mA total current limit.

Lastly, cut the wires that go from the power supply's board to the output banana jacks about halfway. The trick here is that we'll re-route the output wires to be in series with the project's main board. Refer to **Figure 7** which will help make this clearer. Please note that the ground (common) path is not broken, but referenced to both boards. Also note that since the +5V fixed output sources a higher current, the contacts are "doubled up."

The wires are soldered together and insulated with heat shrink tubing. Again, please use color-coded wires to prevent confusion.

Final Tests and Operation

After carefully inspecting your wiring, power-up the unit. After a brief delay, the topmost screen in **Figure 8** will appear. Adjust the variable supplies; you'll see the display changing which is useful to set the proper voltage before actually powering the circuit. Verify that none of the outputs have any voltage with your multimeter.

Now connect a 10 to 20 ohm, three watt load to the fixed output terminals. Press the

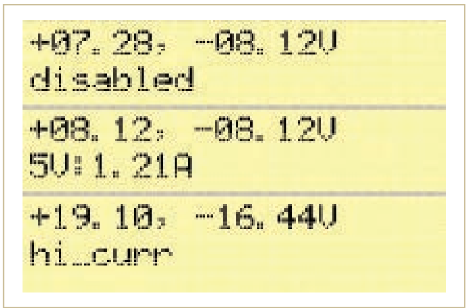


FIGURE 8.

FIGURE 9.

enable button (this is the green button in **Figure 9**, the completed project) and after a brief beep, the message in the middle section of **Figure 8** will appear. The current shown will depend on your actual load. Verify that all outputs have a voltage.

If you have a low valued power resistor that will cause the load to increase above three amps, connect it to the fixed output. The buzzer will start beeping and the bottom section in **Figure 8** will appear. Pushing the button again will revert back to the middle screen and disable all outputs. That's it, you have successfully completed the project!

One final note: Since the fans will not be running continuously, you don't need premium-grade fans. I got mine at a surplus web vendor, for about $6 a pair. However, if your supply does not require the fans, you can omit them, sensor U6, and also header J2 for a savings of about $15. Feel the power! **SV**

Planning Out Your Software for Microcontrollers

By Matt Rusnak

Microcontrollers have certainly gained a lot of popularity in hobby robotics and electronics during the past decade, partly due to their ability to change the function of a circuit without any changes in hardware. This flexibility makes it tempting to write software on the fly, but with a bit of planning, programming can be much more systematic and yield better results.

This is not to say that I think tinkering is entirely bad. It can actually be one of the more creative parts of a project and can lead to a lot of good ideas. Taken too far though, tweaking the software continuously can cause you to rewrite existing code over and over, and may keep your project from ever being finished.

The Project

This article is mainly about a process for developing software, so I have chosen a fairly simple project to highlight its implementation. My hope is that you will take these ideas and use them in your next robot build. It is a puzzle of lighted pushbutton switches that appears in a number of video games, usually as a door lock. The idea is to start with all of the lights turned on; pressing each switch toggles its light and all of its neighbors. To solve it, the player has to turn off all of the lights.

Instead of just building the same exact puzzle, I wanted this version to randomly set up which nearby lights are affected by each switch so that it plays differently each time. It sounds simple enough, but planning brings out a lot of details.

Analysis

There is no single "right way" to plan and develop software for every person and every project. This article explains a way that works for me on small projects. There are a lot of good books on software design methodologies used by professionals, but a simplified process like this may be a good starting point to organize your project's software without making it feel like work.

Most software design methods I have seen use an iterative process, so instead of determining *all* of the needs first and then moving on to implementation, they work in stages, starting with the broadest part of each major feature and testing that they all work together. Then, you work out some details, test the results, and repeat until finished.

A good first step in analysis is writing down what features the project should have. My list was relatively simple:

- Three rows by three columns of pushbutton switches.
- Three rows by three columns of LEDs.
- Pressing a switch should toggle its LED and others.
- Keep one microcontroller pin available to signal winning.
- Use small, lightweight batteries.
- Possibly include a difficulty setting.

Based on these requirements, I determined that the array of switches and the array of LEDs needed six microcontroller pins each. After thinking a little more, I realized I could treat it as a single array of three columns by six rows, using only nine pins. It turns out this assumption was partly wrong, but pretty close — as explained in the next section.

With nine pins for switches and LEDs plus one pin to signal when the user had won (for some visual indicator, sound, or a solenoid to unlock a door like in the games that inspired this project), I would need a microcontroller with at least 10 pins.

Continuing the analysis, since I wanted the game to be set up randomly each time, I would need a single timer. One important detail is that the microcontroller's timer might reset to the same value on each startup, so to make the first game appear random I would have to take input from the user, and use the timer as a seed.

Next, I made a list of the variables I would need, such as the display state, a set of random numbers, temporary variables for switch debouncing, and other items. I like to make columns next to my list of variables showing the minimum and maximum number of bytes for each, and whether each variable is for temporary use or holds its value throughout the program. In this project, I set aside some scratch memory to use temporarily for switch debouncing, processing the display variables, and other uses. This saves memory as long as more than one piece of code does not need the scratch variables at the same time. Knowing how much memory you need is quite helpful in choosing a microcontroller, but remember to allow some extra room for more variables, in case you need to add some later.

Data memory can be tough to estimate exactly when using high-level languages such as C on low- to mid-range microcontrollers with only tens of bytes of memory, since you do not have direct control of which memory will be re-used for local variables. That will probably not be an issue if you use a BASIC Stamp, Arduino, or other more refined device. On small projects, it is not difficult to keep track of a few global variables and carefully reuse them only in short segments of code. This works just as well for most programming languages, but I like to use assembly language on small microcontrollers for finer control.

More About Choosing a Microcontroller

As a hobbyist, I always choose microcontrollers that can be programmed in-circuit and use Flash memory, so they don't have to be removed and erased repeatedly during development. I also look for an internal oscillator for most projects, to avoid using an external crystal or RC network.

There is one important item to point out in the PIC16F505. Section 4.7 of its datasheet explains that when using the CALL instruction or computing an address to jump to, the MCU cannot reach the second half of each memory page, but the GOTO instruction can because it contains an extra bit in its address. In assembly language, you may have to work around this by placing your functions and destinations of computed jumps at specific locations. If you are using a C compiler or Basic, this should be taken care of for you.

Of course, the decision to use a specific part also may depend on having a particular microcontroller on-hand which lets you start building and programming immediately!

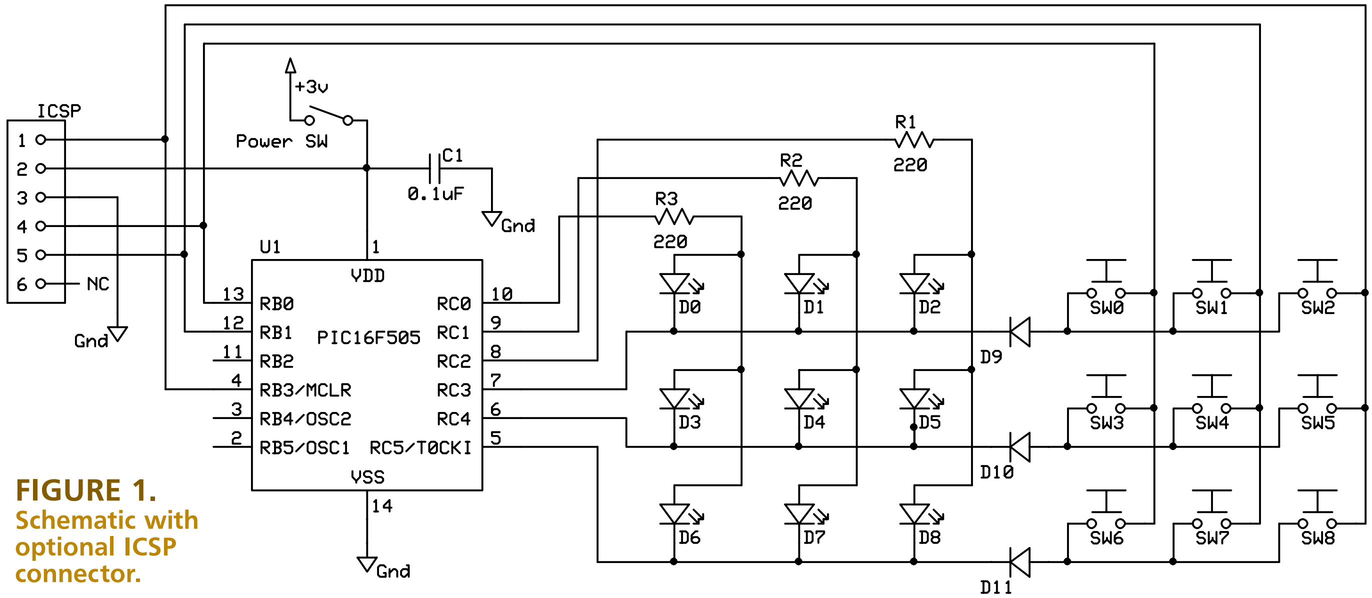


FIGURE 1. Schematic with optional ICSP connector.

Design and Prototyping

In the first pass of design, I chose the microcontroller. During the initial analysis, I found that I would need at least 10 I/O pins and 33 to 52 bytes of data memory, depending on how many temporary variables could be reused, and how many variables smaller than one byte I would pack into each memory location. I also needed a timer to generate random numbers. I do not yet have enough experience to estimate how much program memory I will need, so once I gather a list of chips that fit the other requirements, I'll make my best guess favoring on the high side.

The PIC16F505 fits the requirements, because it has 11 general-purpose I/O pins, 72 bytes of data memory, and one timer. It also has optional internal pull-ups on some of the pins which I did not originally plan to use. This later saved me some extra components. When choosing a microcontroller, it is best to check out its datasheet which can typically be downloaded from the manufacturer's website. I use PICs because I already have an inexpensive USB programmer, and some of the simpler chips can be purchased for less than a dollar each. PICs take more effort to program compared to Arduinos and the like, but once you get used to them, their datasheets can help you choose the best chip for your project. (See the **sidebar** *More About Choosing a Microcontroller* for more details.)

At this point, I usually start a simple prototype by drawing a schematic (**Figure 1**), placing components on a solderless breadboard, and writing some code for the most basic operations to verify that the initial design works. This validates the basic assumptions made during the first pass of analysis, and it is also helps to bring some life to the project. I started with just enough code to test the display, making sure that the LEDs could fade in and out to create interesting effects instead of just turning them fully on or off. This worked well, so I moved on to lighting the LEDs based on which button was pressed.

Remember when I said one assumption was wrong in my initial analysis? I wired the switches so they would share the pulled-up columns with the array of LEDs, but I hadn't noticed that if the user was holding a switch, it would tie that column to ground or to other columns if the user held multiple switches. Holding a switch affected the LEDs in the same column, so I decided to swap the rows and columns, giving the switches their own columns with internal pull-ups and shared rows instead. I also added the diodes (shown in the schematic) to keep the switches from tying two rows of LEDs together if the user pressed two at the same time. Since I hadn't written a lot of code yet while testing out my assumptions, I only had to make minor changes. Even if I had to change the requirements to include more I/O pins and choose another microcontroller, I would not have wasted much time at this point, since the issue was caught early.

After modifying the layout of the circuit, I tested a method for debouncing. I didn't want the program to pause to check each individual switch sequentially, so instead I repeatedly took samples from all of the switches at once and placed them in temporary variables together. I then processed those variables to see if any switch changed since the last stable reading. Even with the cheap switches in my prototype, it seemed to work well enough. For the first prototype, I only tested eight of the nine switches — since the variables are eight-bits wide — to avoid writing too much extra code this early.

Formalizing the Variable List

To finish off the first round of design, it is helpful to make some concrete decisions about how data will be stored. After the most basic features are prototyped, you should have a fairly good idea of how well your assumptions have played out, and whether or not you need to add any major variables. I like to lay out specifically where each variable should be placed in data memory. This is especially important in a microcontroller with banked memory, because switching banks takes extra time and memory. In some microcontrollers, you may not be able to copy data from one bank to another without using temporary variables in the non-banked (or shadowed) memory — unless you switch banks between each item.

I then lay out the memory in a spreadsheet in the same format as the data sheet's register map (**Figure 2**). I use one cell for each memory location, and type in the variable names that I want in each place. This makes them easy to rearrange. When writing your program later, you can easily see which bank holds each variable. To minimize bank switching, the shadowed memory should hold most variables that are used frequently or used by multiple parts of the program. It is also useful to place temporary variables there, so they can be accessed no matter which bank you are using at the time.

	Bank bits			
	00	01	10	11
00h	INDF			
01h	TMR0			
02h	PCL			
03h	STATUS			
04h	FSR			
05h	OSCCAL			
06h	PORTB			
07h	PORTC	Shadowed	Shadowed	Shadowed
08h	Delay0			
09h	Delay1			
0Ah	Scratch0			
0Bh	Scratch1			
0Ch	Scratch2			
0Dh	Scratch3			
0Eh	Scratch4			
0Fh	Scratch5			
10h	Display0	MasksL0	MasksH0	Random0
11h	Display1	MasksL1	MasksH1	Random1
12h	Display2	MasksL2	MasksH2	Random2
13h	Display3	MasksL3	MasksH3	Random3
14h	Display4	MasksL4	MasksH4	Random4
15h	Display5	MasksL5	MasksH5	Random5
16h	Display6	MasksL6	MasksH6	Random6
17h	Display7	MasksL7	MasksH7	Random7
18h	Display8	MasksL8	MasksH8	Random8
19h	AppState			Random9
1Ah	SwState0Prev			
1Bh	SwState0Curr			
1Ch	SwState1Prev			
1Dh	SwState1Curr			
1Eh	GameOptions			
1Fh	OptionCounter			

FIGURE 2. Data memory diagram with variable names.

PUZZLE BOX PARTS LIST

Specific Parts, Supplier & Part #
Microcontroller
Microchip – PIC16F505
Pushbutton switches (9)
RadioShack 275-1547 or 275-1571
LEDs, D0-D8
RadioShack 276-1622 (assorted)

Common Parts
Power switch, SPST
0.1 µF capacitor, any type
Resistors R1-R3, 220 ohms
Diodes D9-D11, general-purpose such as 1N4001
Single-row six-pin header for ICSP (optional)

*Download files that accompany this article are available at **www.servomagazine.com.***

Additional Analysis

Analysis and design tend to blend together, and that is okay. It would be hard to avoid making any decisions on the design while analyzing the requirements or the results of testing your design. Forcing yourself to stop and analyze where you stand is a good way to check if you are really designing the software the best way you can.

In the prototype, I had stored the entire value I read from PORTB as a variable to use each time I ran through the main loop. This wastes some memory, but more importantly it makes it harder to work with the switch values. I noticed at this point that it would be easier to store the previous and current values of each switch as a single bit in two separate variables. If these variables are XORed together, a bit is set for each switch that changes states.

The display and switch reading code in the prototype worked well, so I determined there was nothing major to change there. I tweaked the variable list (adding the new previous/current variables for the switch states) and after reviewing the variable list again, everything looked good, so no additional analysis was needed.

Matt Rusnak can be reached at mrusnak@roadrunner.com.

Continuing Design

At this point, I often prefer to set aside my prototype's code (still saving a copy), and start over with a blank file. When I first heard that some software developers work this way, I thought it was crazy.

Since my prototype was quickly cobbled together without exhaustive planning, starting over this early in the process can actually lead to simpler code with fewer workarounds caused by design changes. Of course, some of the prototype code can be reused if it already works well.

There are many ways to set up the main loop of a program. For any project that has multiple modes of operation, I like to make the main loop run all of the common parts — such as the display and input in this project — and then call a different function depending on a single variable that represents the current state, or a high level representation of what the device should be doing.

This means that the code for any state can use the values of variables that are maintained by the common code. The current state's code will run for each pass of the main loop, and that code decides when to change to another state.

For example, the startup sequence of this project sets a pattern for the display and checks the variables representing the switches, but it lets the main loop actually write to the display and read the pins connected to the switches.

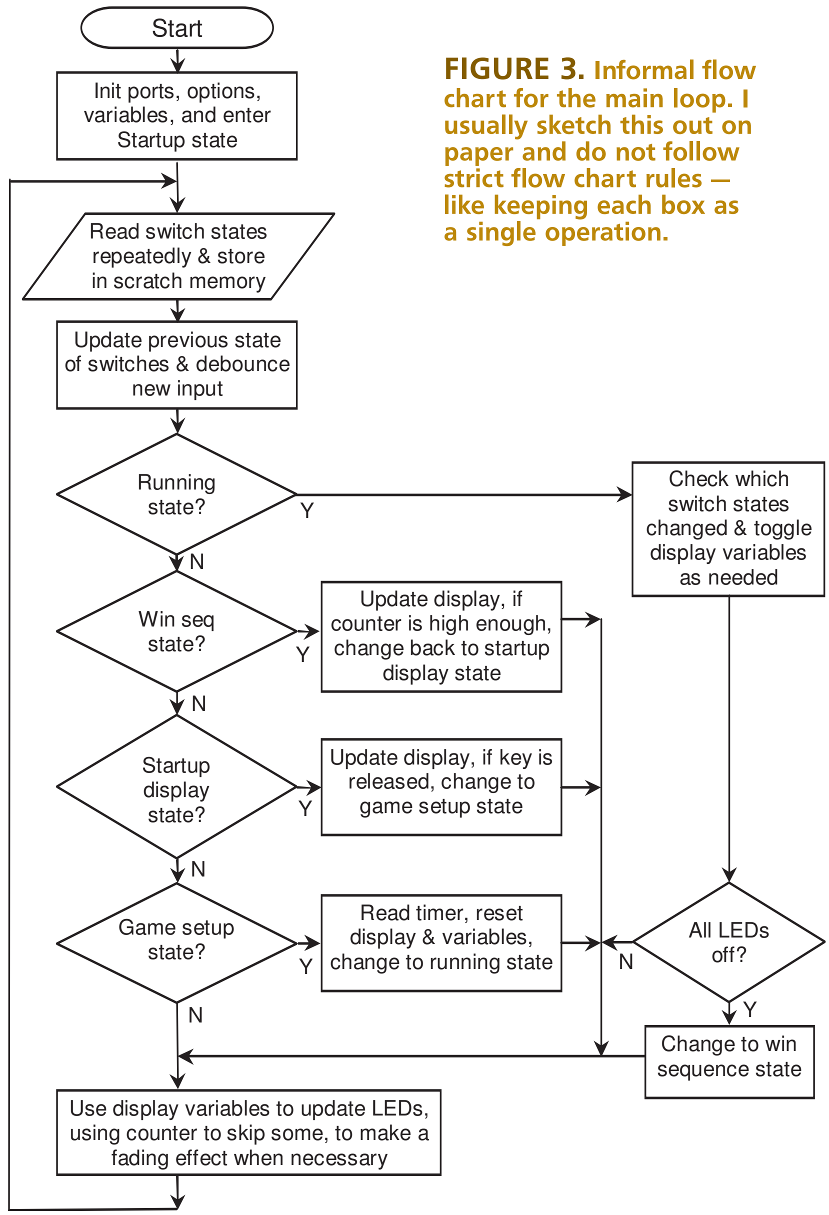


FIGURE 3. Informal flow chart for the main loop. I usually sketch this out on paper and do not follow strict flow chart rules — like keeping each box as a single operation.

When a button is pressed, the startup sequence code changes the state variable to the next state which sets up the display randomly. That state only runs once and then changes the state variable again to the "running" state (which handles the game logic until the user wins).

Designing the main program can be overwhelming, but breaking out the major states can help to simplify it. If you are familiar with flow charts — even in the informal way that non-programmers use them — it can be helpful to draw out a high-level view of how the software will work. (Check out the example in **Figure 3**.) Or, writing out a list of what the main program should do on paper and drawing arrows to show loops or decisions can help in planning.

After you have the basic flow drawn out, writing the code to implement it becomes more mechanical, so there is a lower chance for errors than coding on the fly. If you come to a point where it is unclear what to do next, it can help to just write out some instructions and step through what should happen in different situations.

Feature Creep!

When designing software, it can be easy to get sidetracked and start adding unplanned features. It happened in this project when I decided to add a "menu" for difficulty settings, display options, and a demo mode. When I realized I was getting too far from my original plans, it was a hard choice, but I put some of the other features on hold.

I did add the menu — which was fairly easy since it was just another state to add to the main loop — but I only implemented one option in this version of the puzzle. The option I kept was the basic difficulty setting, so the user could choose to play the default random game or the original version where each switch affects its neighboring LEDs.

It is obviously okay to add more features if they are important to you, but if it goes too far, it is possible that the changes will complicate the existing software. This could make it take longer to write or debug. At its worst, feature creep can cause the project to go into that "in progress" box that many of us seem to have. I am happier to have finished the first version of this project, and one of these days I may work on the additional features.

Conclusion

This puzzle box was just a basic example, but the concepts can be applied to any device or subsystem. For the next version, I still may add the difficulty options, and I also plan to make the puzzle more truly random, even though I haven't noticed any obvious patterns yet. I also didn't implement the pin that does something when the user has won, and instead just made a flashy display for now.

I may need to choose another microcontroller unless I can optimize some of the code better, since I ran lower on program memory than I originally expected. More feature-rich microcontrollers would prevent this problem (if you have the funds and physical space for them) and they can simplify software design as well, since they have fewer limitations.

I hope this article gives you some ideas about a basic method of software design you can use. This general process is only a guideline for getting started, but even a little planning can go a long way. You can add more rounds of analysis and design based on your project's needs, and this process can be adapted to the way you prefer to work, even incorporating other design methods you have seen. **SV**

This article shows how you can implement PWM in your own robotics projects. By building these two simple devices — a general-purpose, low voltage light dimmer and a fan motor speed controller — you will clearly see the principles in action.

Controlling Motors and Lights With Pulse Width Modulation

By Jürgen G. Schmidt

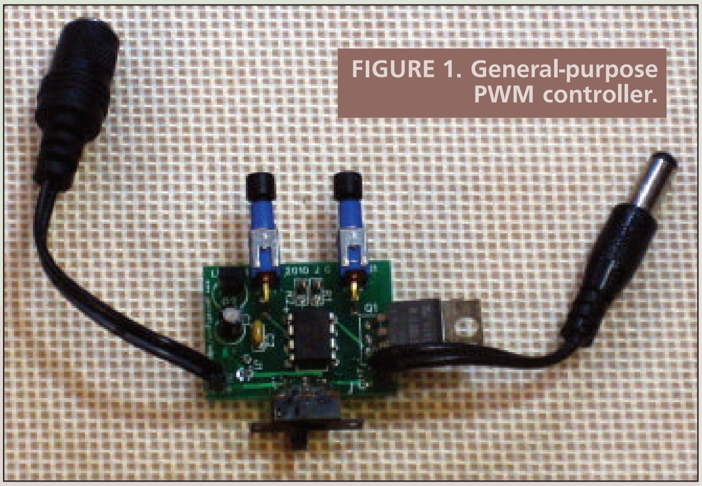


FIGURE 1. General-purpose PWM controller.

Background

There are switches everywhere! Most of them turn things either on or off, but sometimes we want something turned only partly on. Our cars are an example of this. We use an accelerator to control the engine speed. A simple on/off switch for the engine might be exhilarating but ultimately not very useful. The same is true for some of our lights, fans, and robots. We need to control how much they are turned on.

I had just finished replacing my halogen workbench light with an LED light strip I made from three watt LEDs. Now, I needed a dimmer for it. The LEDs work best at a constant voltage, so a PWM–based control seemed like the way to go. PWM (Pulse Width Modulation) consists of turning a device on and off very rapidly. The proportion of the time that the device is switched on is determined by the width of the pulses; therefore the term pulse width modulation. You will find PWM used in many projects, particularly with robot motors.

A simple alternative to PWM is to use a variable resistor in series with the device you want to control. This wastes a lot of power as heat, and as the voltage drops the device you are controlling may stop working too soon.

The 555 timer chip is frequently used to generate PWM signals. This chip — along with a few resistors and a capacitor, some calculations, and experimenting — will yield an inexpensive and effective result. You can see the basic astable circuit description and calculations in the 555 datasheet (see **References**). However, I wanted a design with pushbutton controls for increasing and decreasing the speed or brightness, and I needed the flexibility to easily experiment with the various parameters of a PWM signal. I prefer to do this in software rather than hardware.

I use PIC microcontrollers in many of my projects, frequently as a replacement for discrete components. In this case, I selected the PIC12F683 since it has a hardware PWM component. My first PWM controller was incorporated into the overall design of the project but I soon found that I could use PWM for all sorts of things. This led to the design of a general-purpose PWM controller that could be inserted between a power supply and a device — usually a light or a fan.

The PWM controller ended up as a parasitic system. By that, I mean that it does not need a separate power supply. Instead, it taps into the device that is being controlled. I built my first one between male and female 2.1 mm power connectors (**Figure 1**). I tried this out on some of the 12 volt fans I have on my desk and workbench. Previously, in order to reduce noise and avoid blowing papers away, I had adopted various strategies for slowing down the fans. These included using two fans in series and using six volt power supplies for 12 volt fans. A general-purpose PWM controller would give me consistent and variable control of my fans.

Design and Testing

In the course of designing and testing this general-purpose PWM controller, I learned a few things. Backing up a little, a PWM signal has two key parameters: a period and a duty cycle (**Figure 2**). The signal is a square wave with the bottom at zero volts and the top at the driving voltage. In the case of my fans, this is 12 volts. The ratio of the width of the top section to the width of the bottom section is the duty cycle. If they are both the same length, then the duty cycle is 50%. If you make the top wider and the bottom narrower, then the duty cycle increases and the fan turns faster or the light is brighter, whichever the case may be. The total of one off and one on cycle is the period. If the period is too long, you will see the light flicker. In the case of the motors, they will hum or squeal at certain frequencies. Selecting the proper period or frequency (periods per second) is important for a successful PWM controller.

The circuit for the PWM controller is fairly straightforward (**Figure 3**). Parts selection is not critical — you probably have most of the parts on hand. The power to the target device is interrupted by a power switching transistor such as a TIP31 or MJE3055t in a TO-220 package. This will introduce a drop of about half a volt between the input and output side. If you use a Darlington transistor such as a TIP120, the voltage drop will be slightly higher, but still under one volt.

Figure 4 shows how to connect a TO-220 NPN transistor. A 78L33 or 78L05 voltage regulator steals some power to supply the 12F683 which, in turn, provides a PWM signal to the transistor. Bypass capacitors C1 and C2 provide additional regulation. Input voltage to the overall system is limited by the specifications on the power transistor and the voltage regulator. The MJE3055T and TIP31, for example, are rated at 60 volts. The 78L33 and 78L05 voltage regulators, however, have a maximum input of 30 volts. The unregulated 12 volt transformers for my fans actually put out around 16 volts and the LED power supply I use puts out less than 30 volts, so I'm safe with these.

Two pins of the microcontroller are connected to pushbuttons that connect to ground. The pins are held high with pull-up resistors — anything from 4.7K to 100K will work. The pushbuttons increase and decrease the percent activation. I designed a 1 inch by 1-1/2 inch printed circuit board (PCB) to hold all the parts, along with an optional power switch. The desired power connectors can be added on the ends of the PCB or it can be wired inline with a project. For space reasons, I did not include a programming connector on the PCB or the breadboard. Instead, I used an eight-pin IC clip to avoid repeatedly pulling and inserting the chip during testing (**Figure 5**).

Since I had spare pins on the chip, I also connected one to a heartbeat LED and another to my PC serial port to get debugging output from my breadboard setup. (See the **sidebars** on Heartbeat LED and Terminal Programs.) This allows me to monitor the actual duty settings and their effect on the fan and light.

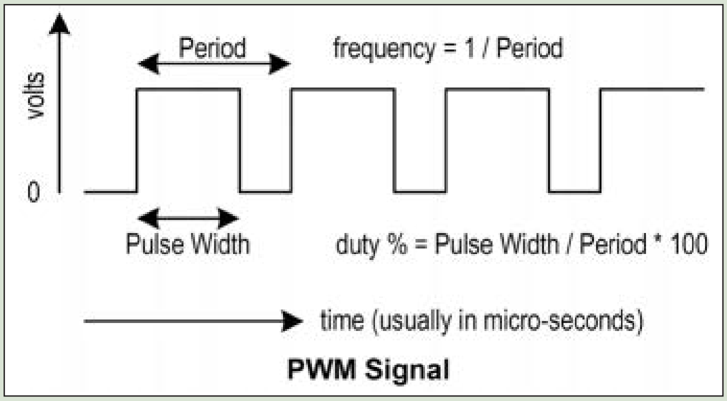


FIGURE 2. PWM signal diagram.

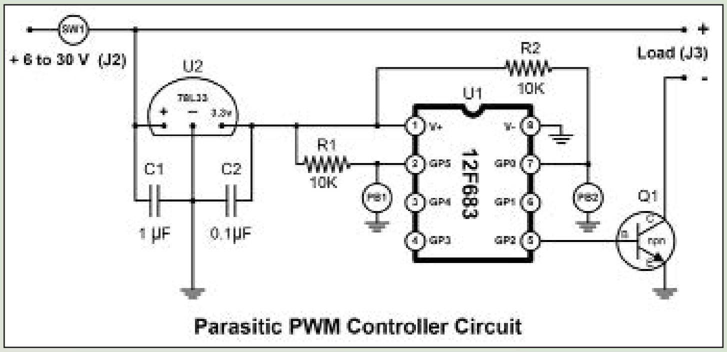


FIGURE 3. Circuit diagram.

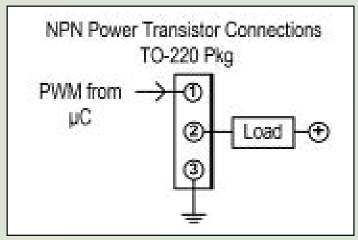


FIGURE 4. T220 NPN power transistor connections.

Software

The initial software was written two years ago for the PICBASIC PRO compiler. This has a built-in HPWM function that requires three parameters: PWM channel, duty, and frequency. This program is shown in **Figure 6**.

My current microcontroller projects include handling multiple interrupts, as well as communicating with the Internet. PICBASIC isn't really suitable for this, so I recently switched to the Custom Computer Services (CCS) C compiler. While this change gave me additional flexibility, it also gave me additional complexity. Instead of a single PWM function, I now need to use three functions. One of these — the **setup_timer2()** function — requires some careful calculations to arrive at the right parameters for the desired frequency and duty. I had to read the datasheet and application notes several times to make sense of them.

To help in calculating the correct parameters for the CCS C PWM statements, I developed an Excel spreadsheet that provides the actual CCS C statements to include in the program. It has two boxes of calculations. The top box is for determining the results of various parameters using the

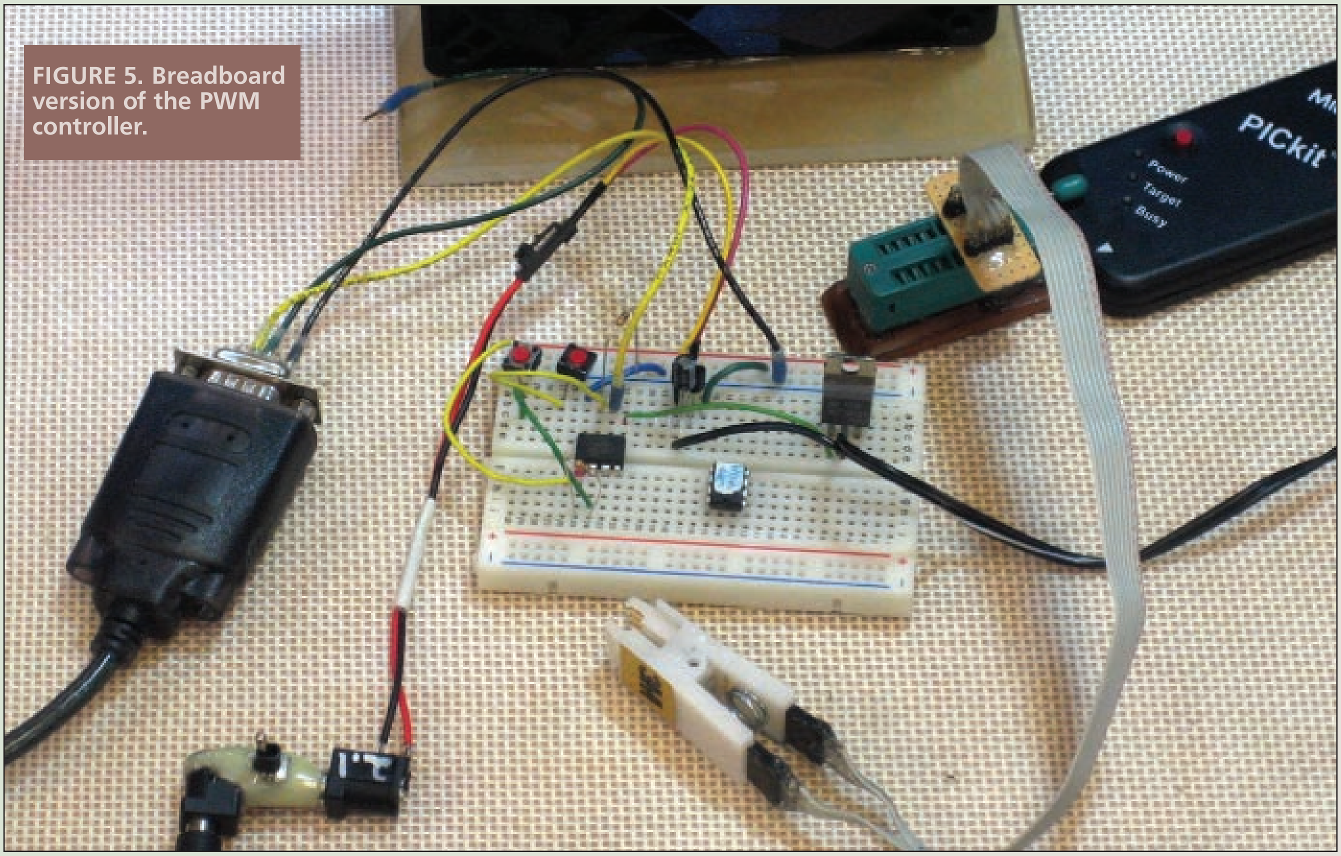


FIGURE 5. Breadboard version of the PWM controller.

HPWM calculations Excel spreadsheet.

Timer2 Calculations for PIC12F683 PWM
For CCS C Compiler

Miscellaneous Calculations

Crystal Frequency (Fosc)	8 000 000.	Hz	
Prescaler (1, 4, 16)	1.		
Cycle Time (Tosc)	0.000 000 5	Sec	2 000 000. Hz
PR2 (Timer2 period: 0-255)	200.		
PWM Period (overflow)	0.000 100 5	Sec	9 950.2 Hz
Duty Resolution	804.	←	1024 is maximum possible (10 bits)
PostScaler - sets interrupt (1-16)	16.	←	Not used for PWM
Interrupt period	0.001 608	Sec	
Crystal	8 000 000.	←	Enter crystal frequency (Hz)
Desired Period Frequency	16 000.	←	Enter desired PWM frequency

	PR2	Duty Range
Prescaler = 1	124.00	0 to 500
Prescaler = 4	31.00	0 to 128
Prescaler = 16	8.00	0 to 36

```
setup_timer_2(T2_DIV_1, 124, 1 );
setup_ccp1( CCP_PWM );
set_pwm1_duty( 250L );       // square wave output - 50% duty
```

formulas provided in the Microchip documentation. The bottom box accepts two values: a microcontroller oscillator frequency and a target PWM frequency. For this project, the oscillator frequency is the internally generated 8 MHz. I tested several computer fans — small and large — at different frequencies and discovered that a PWM frequency of 16 kHz worked best for most of them.

At this frequency, there was minimal humming or squealing. If you call up the spreadsheet, you will see these values entered. If your particular fan is noisy (especially at lower speeds), experiment with different frequencies. C statements are generated that yield the highest resolution or number of steps for the duty range. In this case, all three values for the Timer2 prescaler are allowed. If you change the target frequency to 4 kHz, you will see that one of the prescaler values is not allowed. For a detailed explanation of this, see the Microchip datasheets and application notes listed in the **References**. The spreadsheet is available on my website at **www.jgscraft.com/ledpwm** or the *SERVO* site at **www.servo magazine.com**.

In the course of converting the original PICBASIC program to C, I came

```
'—— 683PWMLED.bas ——
' PWM LED Dimmer
' Jurgen G Schmidt
' Target:          PIC12F683
' Compiler:        PICBasicPro 2.50b
' IDE:                    MicroCode Studio
'-- internal clock
'-- disable MCLR
@ DEVICE INTRC_OSC_NOCLKOUT, MCLR_OFF
DEFINE OSC 8
OSCCON = $70                '8mhz for internal
ANSEL = 0                   'All Digital
CMCON0 = 7                  'Comparators off

'—— Port Assignments & Variables
pinHi   var     GPIO.0      'Brighter
pinLo   var     GPIO.5      'Dimmer
duty    var     BYTE

'—— Initialize System
read 0, duty                'read EEPROM

MainLoop:
    hpwm    1,Duty,24000    'GPIO.2 at pin 5
    pause 200
    if pinHi = 0 then gosub Brighter
    if pinLo = 0 then gosub Dimmer
    goto MainLoop
end
'————————————————
Brighter:
    if Duty => 255 then return
    Duty = Duty + 15
    write 0, duty    'save to EEPROM
    return
'————————————————-
Dimmer:
    if Duty = 0 then return
    Duty = Duty - 15
    write 0, duty    'save to EEPROM
    return
'————————————————-
'—— end of 683PWMLED.bas ——
```

FIGURE 6. PICBASIC PRO listing.

up with some improvements. The original program saved the speed or brightness setting in EEPROM so that when I turned the fan or light back on, it would start at the same level. The problem with fans is that you can slow them down to a 10% level when they are running, but you can't start them at that level. I added a feature that would boost the initial fan speed if it was below a certain level to get it started, and then drop back down to the slower saved speed. This resulted in two versions of the program: one for lights that did not have the "boost" feature; and one for

Heartbeat LED

When I first started working with microcontrollers, I would frequently chase software bugs that were in reality hardware issues. Now when I prototype, I ALWAYS include a heartbeat LED on the breadboard. If I'm working with a development board — which usually has one or more LEDs — I make one of them the heartbeat. I never take it out until I'm finished. Blinking an LED at the beginning of an embedded program (and throughout) verifies that your system is alive. The code is very simple; just a few lines to turn the LED on and off. Get this working first and then later on if the LED is not blinking, there is probably something wrong other than your code. Once I implemented the heartbeat LED consistently, I've saved myself considerable aggravation. Where initially I suspected my code, I discovered that batteries had depleted, programming or prototyping wires had come loose, a critical component had been harvested from a breadboard for use elsewhere, and so on. These are simple, silly things that are easily fixed, but if overlooked can make you doubt your sanity. If the LED does not blink on power-up, I check the hardware and environment before I mistrust my code.

Using a heartbeat LED and writing the universal microcontroller equivalent of "Hello world" is also useful when starting with new hardware. It verifies that you have a viable configuration. It can also provide feedback on the correct oscillator or crystal settings.

Once I have the heartbeat, I add the serial output routines for more detailed diagnostics. Then it's on to the rest of the project. Once you have the heartbeat and serial routines under control, you have a good framework for proceeding.

Terminal Programs

For debugging, I connect my PIC projects directly to a terminal program running on my development PC. I have not had any trouble connecting an output pin from my microcontrollers directly to a PC serial port, even at 3.3 volts. I just have a ground wire to pin 5 and another from the serial output pin to pin 2 on the DB-9 connector. With the disappearance of physical serial ports on PCs, I'm usually connecting to a serial-to-USB cable, such as shown in **Figure 5**.

Microsoft Vista and Windows 7 do not include a serial terminal program. If you are still using Windows XP — as I am for my development systems — then you can use Hyperterm. I find it a nuisance to work with and have found some free alternatives that work well on all systems. Aside from the simplicity of use, some also support TCP and UDP which makes them handy for testing TCP/IP communications programs. My favorites are:

The Hercules Setup utility from HW Group (**www.hw-group.com/products/hercules/index_en.html**).

TeraTerm Pro is available from **www.ayera.com/teraterm/**.

NetBurner has utilities for monitoring and debugging communications. These are available from **www.netburner.com/support/public_downloads.html**. The serial terminal program is mtty.exe.

None of these programs require installation — they run straight from the exe so they are easy to carry around and use. Some of them (Hercules and TeraTerm) do not list the available COM ports; that is, you have to know ahead of time which COM port you want to use.

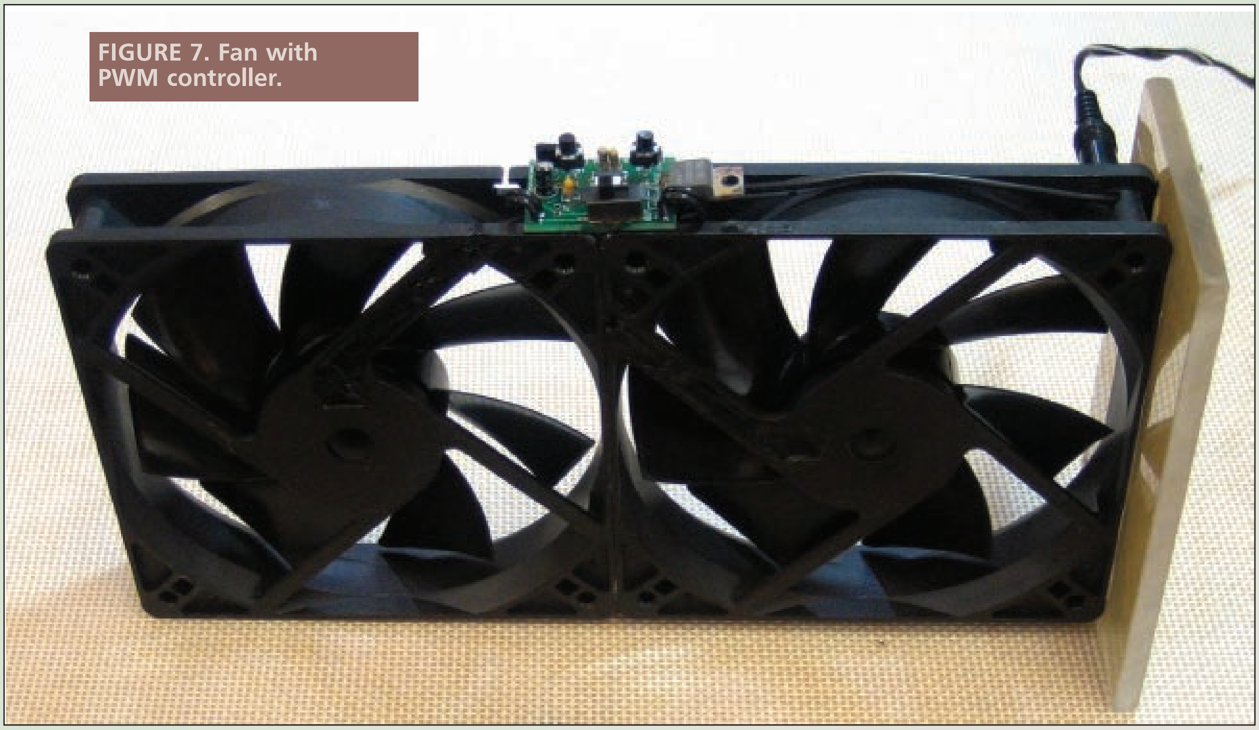


FIGURE 7. Fan with PWM controller.

fans with the boost feature.

The latest version combines both features into a single device. There is a setup routine that sets the controller to fan or light mode. If you hold down the "increase" button during power-up, the controller is put into light mode. If you hold down the "decrease" button while turning on the power, the controller is put in fan mode. This setting is saved in EEPROM. Since I use these PWM controllers as parts of other projects, I only need one version now that I can easily switch back and forth, depending on what I am working on.

With the appropriate hardware, this code can be adapted to a variety of uses. When driving a robot, for example, you can gradually ramp up the duty value in a loop so that the wheels don't spin from the sudden full-on activation. If you use it to control lights, you could gradually turn them on and off.

When you look at the C code, you'll see statements bracketed by **#ifdef DEBUG** and **#endif** statements. The serial output and heartbeat LED statements are enclosed in these so they can be disabled easily in the release version.

Finished

Once I knew the circuit would work, I sent the Gerber files to Silver Circuits for manufacturing the boards. I refined the software while I waited for the boards to come back. The board has a position labeled J1 which is where the incoming power is connected if you are not including a switch in the design. If you are including the switch, external connector J2 is connected to the pads marked "IN." Power to the load — via external connector J3 — is connected to the pads labeled "OUT." The square pads are positive. Please note the circuit was primarily designed for controlling low voltage lights and brushless computer fans. There is not any built-in protection against transients or surges coming in from the load.

Figure 7 shows the controller attached to the side of a pair of fans. I can now easily adjust the speed on these from "Hurricane" to "Mild Breeze." **Figure 8** shows the controller integrated into my under-the-shelf LED light bar. This photo only shows the first three of the eight LEDs that are

Figure 8. Workbench light with PWM dimmer.

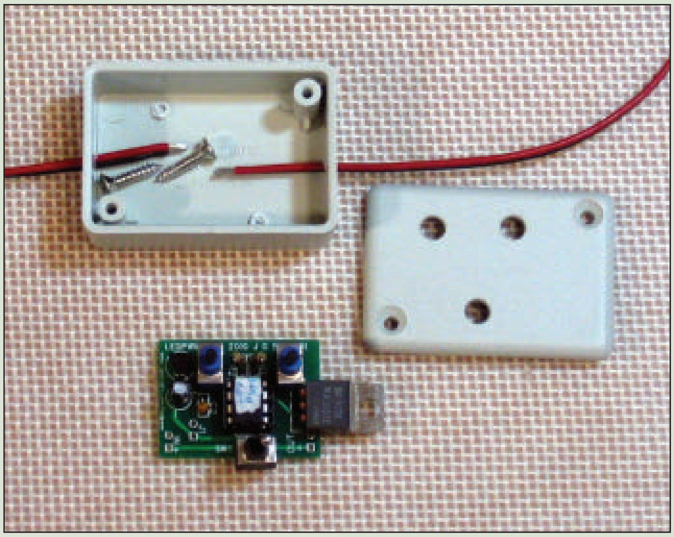


Figure 9. Light controller before assembly.

mounted on a strip of aluminum with heat-conducting epoxy. This strip is, in turn, attached with spacers to the underside of the shelf so air can circulate on all sides of it. High power LEDs require heatsinks to dissipate the heat they produce. You can see the completed circuit board and drilled box before final assembly in **Figure 9.** If you compare it with **Figure 1** and **Figure 7**, you can see that the PCB supports switches in different configurations.

Once you have worked with this basic PWM circuit, you will be ready to implement PWM in larger projects that will include such things as driving robot wheels. Unlike the brushless fan motors I have been using here, drive motors will usually require additional electronics to address direction of rotation and the reverse EMF that is generated when they get turned on and off. A PWM signal will generally be used to control the speed of the motor. **SV**

Parts List

Designator	Component	Source/Part Number
PCB	N/A	Silver Circuits, N/A
U1	PIC12F683	Mouser.com, 579-PIC12F683-I/P
U2	78L33	Mouser, 511-L78L33ACZ
Q1	MJE3055T	Mouser, 511-MJE3055T
PB1, PB2	Pushbutton switch	All Electronics.com, PB-157
R1,R2	10K 1/8 watt	
C1	1 µF 50V	
C2	.1 µF 50V	
J2	2.1 mm jack (power in)	All Electronics, DCJ-1
J3	2.1 mm plug (power out)	All Electronics, DCSID
SW1	Any suitable slide or toggle switch (optional)	

The spreadsheet, hex, and source code files are available at **www.servomagazine.com** or **www.jgscraft.com/ledpwm**. PCBs and pre-programmed chips for this project are also available.

Jürgen Schmidt can be contacted at jurgen@jgscraft.com.

References

Fairchild LM555 Datasheet
www.fairchildsemi.com/ds/LM/LM555.pdf

PIC12F683 Datasheet
www1.microchip.com/downloads/en/DeviceDoc/41211D_.pdf

Microchip Application Note AN594: Using the CCP Module(s)
www1.microchip.com/downloads/en/AppNotes/00594B.pdf

Microchip Application Note AN564: Using the PWM
www1.microchip.com/downloads/en/AppNotes/00564b.pdf

PICBASIC PRO Compiler
www.melabs.com

CCS C compiler
www.ccsinfo.com

PCB Manufacturing
www.silvercircuits.com

By Daniel Ramirez

Using a VEX Controller

The Great VEX LCD Display Experiment

An LCD display is a very useful device that is widely used in consumer electronics, commercial, industrial, and automotive applications to display menus, status information, and diagnostic details. LCDs first became widely available when they were used to display alphanumeric information in calculators, alarm clocks, and watches.

There are two types of LCD displays: the newer graphics displays and the classic character LCD displays in either monochrome or color. The new OLED bright color LCDs are also gaining popularity. Learning how to use these versatile output devices is a must for anyone involved in electronics.

This article will show you how to connect a low cost character LCD to a VEX microcontroller. There are two methods of interfacing LCDs: one is to use a parallel bus and the other is to use a serial interface such as RS-232, SPI, or I²C. In this experiment, the LCD is connected via a parallel bus which is used to send data to the display. This is the easiest method to use and understand.

An LCD display for a VEX microcontroller is also useful for debugging and testing embedded applications used in robotics since it allows you to display data in debug output statements using the printf function. An LCD display also provides a convenient method of displaying text and status and error messages to the VEX user via a portable User Interface (UI). When this is combined with a keypad, it provides visual feedback from the embedded VEX microcontroller that is not connected to a PC (or laptop) when running in autonomous mode. This makes it a very portable diagnostic tool, especially during contests or in the school lab environment. In one evening, you can be displaying data from your VEX controller with your own homemade LCD application.

Character LCD displays such as those shown in **Figure 1** are widely available in 16, 24, and 40 character formats with one to five lines. They can easily be found on the Internet or at surplus electronics stores for relatively low prices. In order to display more characters per line, higher resolution displays (such as the large blue one shown in **Figure 1**) are needed that can easily show up to 80 characters. Some styles can also draw geometric figures such as lines, boxes, circles, and triangles in monochrome, and newer LCDs can draw text and graphics in color. These kinds of displays are often found in vending machines, bank terminals, commercial, and consumer electronics devices, and other embedded controller applications. Some of these LCDs have a simple serial RS-232 interface that can be

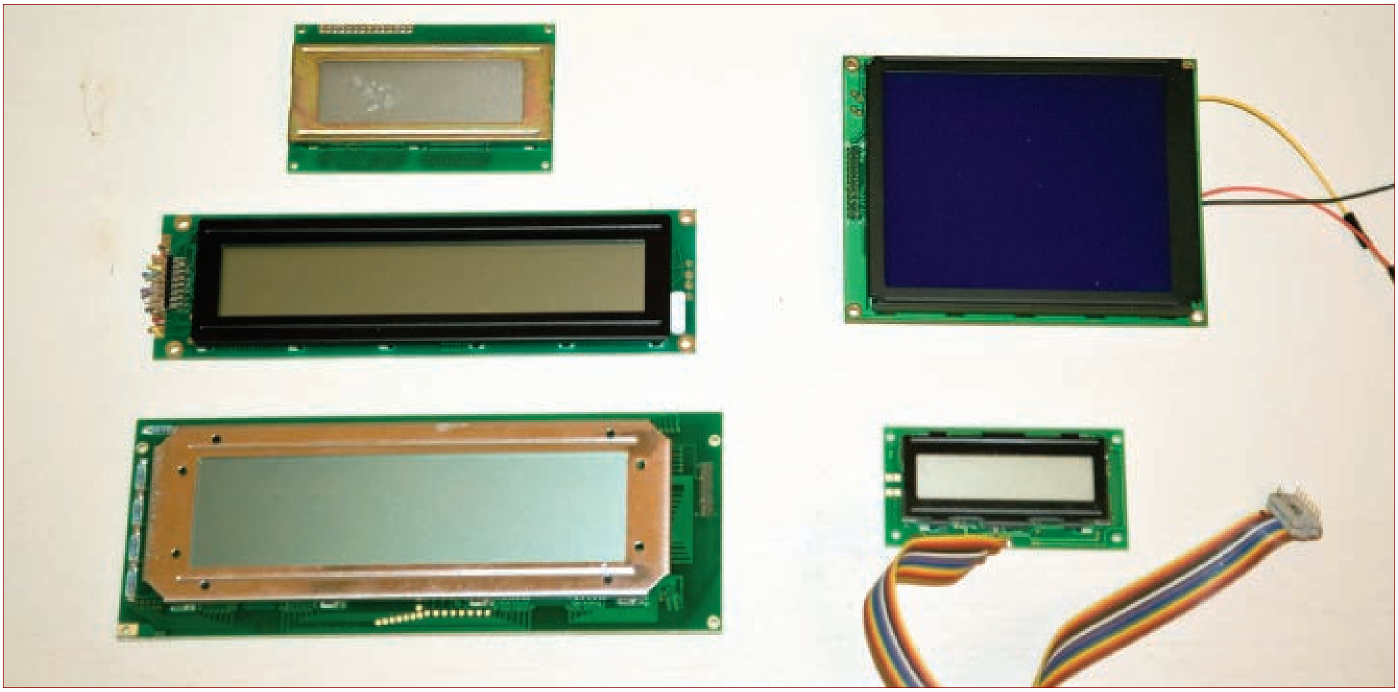


FIGURE 1. An assortment of character LCD displays that are widely available in 16, 24, and 40 character formats with one to five lines that can easily be connected to a VEX microcontroller. The large blue high resolution LCD that can easily be connected to the controller's UART.

connected to the VEX controller's UART.

VEX Character LCD Display Applications

The character LCD display can be used by hobbyists to display sensor data, such as: distances to objects reported back from the VEX ultrasonic ranger; accelerometer values; and battery voltage levels. It can also be used for displaying accelerometer readings, tilt angles, battery voltages, temperature, sensors, or even the latest encoder counts returned from a particular motor connected to a VEX encoder. It can be used to display messages (like "Hello World") and commands sent to a VEX robot (see **Figure 2**).

Another advantage to using an LCD is that they don't consume much power from the microcontroller. This helps the robot application run longer as

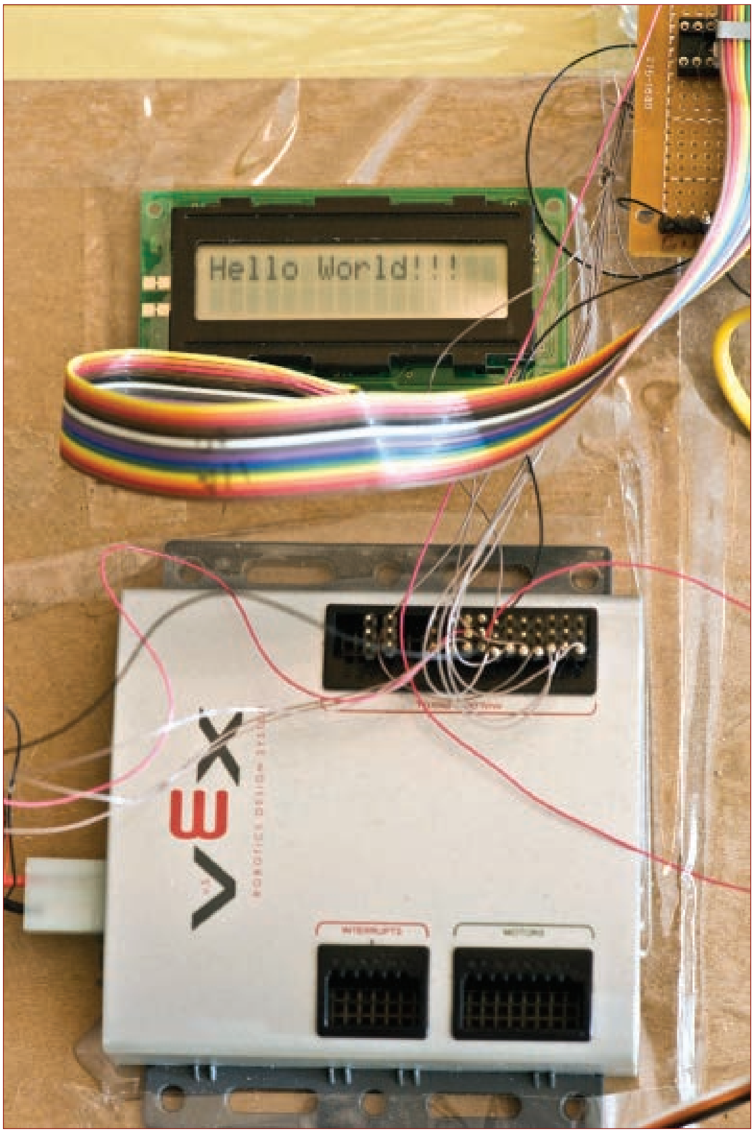


FIGURE 2. An LCD can be used to display sensor data such as distances to objects reported back from an ultrasonic ranger. The latest encoder counts returned from a particular motor connected to a VEX encoder, warning messages to the operator, and commands sent to a robot.

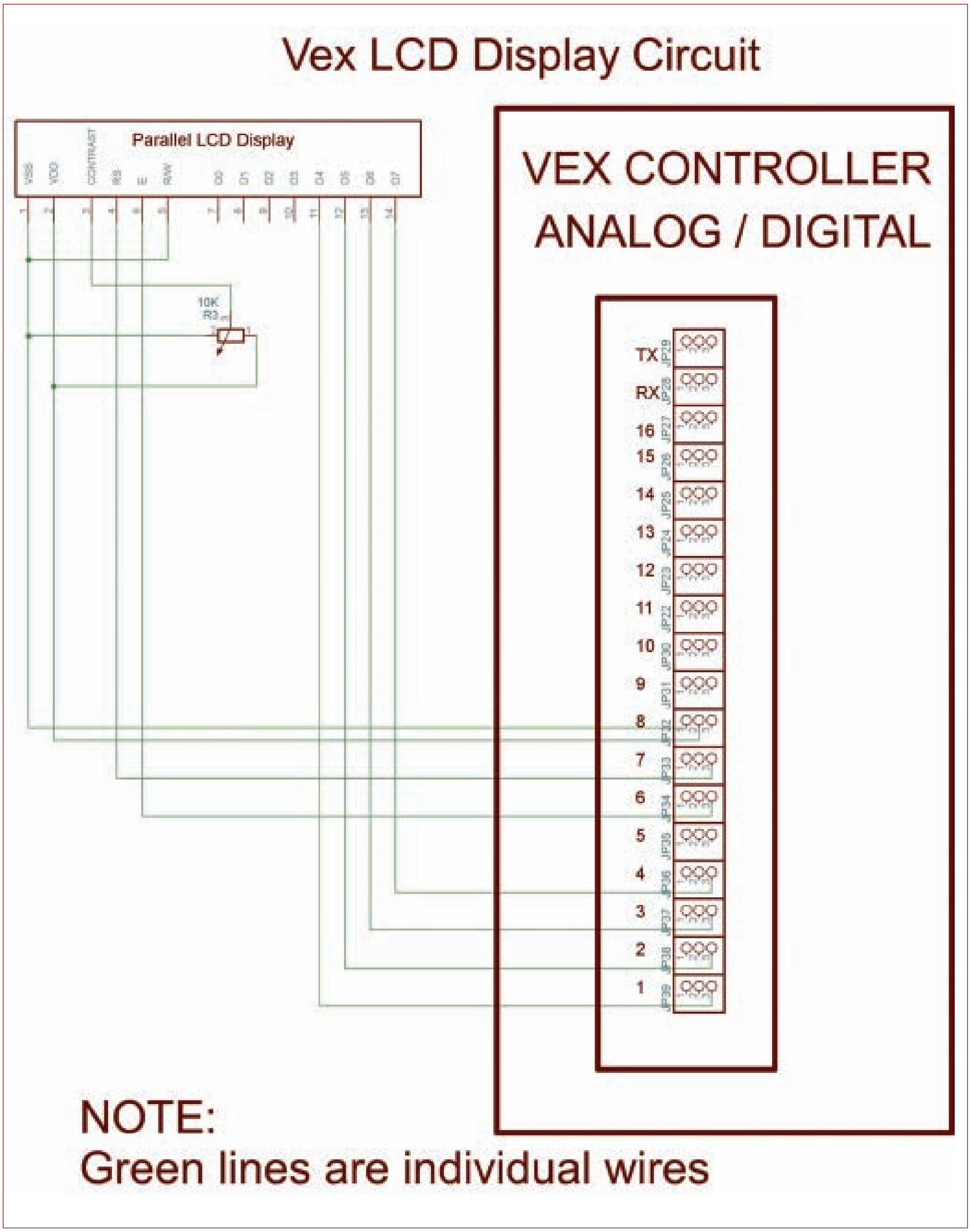


FIGURE 3. My DIY HP-45 calculator using two MAX7219 ICs to make this colorful 10-digit numeric LED display by just wire-wrapping each LED segment to the MAX7219 driver.

compared to a numeric LED display that we covered previously in these articles.

Another novel use for an LCD is to obtain status information remotely with tele-operated VEX models. This can be accomplished by using the VEX RC and wireless video camera to take advantage of an LCD display placed in the camera's field of view to visually monitor the robot status from a distance.

We will talk about how to interface a low cost LCD display and program it using PIC18 C, but for those readers who want to build it to simply see how LCDs work, just follow the simple instructions given here to build the four-bit parallel bus LCD display. Although it uses more I/O pins than the IFI offering, it costs less ($14.99 from **SparkFun.com**).

TABLE 1. Subset of the commands that can be sent to the standard Hitachi HD44780 LCD controller.

Command Name	Value
LCDCMD	254
LCDCLS	1
LCDHOME	2
LCDLINE1BASE	128
LCDLINEINCR	64
LCDSCRLRIGHT	28
LCDSCRLLEFT	24

Using an LCD Display for Debugging VEX C Applications

Debugging and testing VEX C applications can be difficult when not connected to a computer since the only clues to the robot's bad behavior is its movements or lack thereof. In this case, carefully placed printf messages to the LCD can provide some insight into what has gone wrong by displaying specific diagnostic information, including the latest motor commands that were sent to it and the latest sensor readings.

Be sure to add appropriate pauses between lines of text sent to the LCD by using the pause function provided. This way, the user gets a chance to read the message before it's overwritten by the next message.

Interfacing the LCD Display to the VEX Controller

In order to build the low cost LCD display described here, you will need to order a parallel two line by 16 character module. SparkFun or **AllElectronics.com** have them for around $10-$15. The LCD module selected must use the standard Hitachi LCD controller, otherwise it will not work with the firmware provided in this article (go to **www.servomagazine.com**).

The LCD's parallel interface has a 14-pin connector, but we will end up using only 10 of its pins. The LCD construction is easy and very economical since the part list is minimal. The VEX microcontroller does most of the work as shown in the schematic in **Figure 3**.

The LCD module is connected to the microcontroller using a four-bit parallel bus (only nine of the 16 digital input/output pins are used). This still leaves seven digital I/O pins, the interrupt pins, and the motor control pins for use with other VEX devices. Some LCDs provide a backlight that requires a separate five volt power supply, so that increases the connector pins to 16.

The Hitachi HD44780 LCD controller is very flexible and provides functions to position the cursor, clear the display, and change the text attributes to blinking, bold, backspace, scroll, or reverse video, etc. The commands that control the Hitachi controller are shown in **Table 1**. The lcd.c application that I provide with this article can easily be modified to handle other LCD display models such as a one line by 24 character display or a four line by 20 character display, as long as it uses the Hitachi controller.

Usually, you can find markings on SMT ICs on the back of the LCD indicating the model number and manufacturer. Some LCD displays provide pin headers, but most just provide thru holes with .100 inch spacing. If this is the case, then I suggest soldering .100 pin headers to it for a convenient hookup to the microcontroller.

When working with LCDs, I have found previous *Nuts & Volts* articles written by Scott Edwards and Jon Williams for the Parallax BS2 and BSX to be an invaluable resource. In fact, some of the LCD code was ported from their Parallax PBASIC examples to PIC18 C and modified to run on the VEX controller.

These commands are sent via the four-bit parallel bus using the WriteLCDCommand routine or WriteLCD routine to perform various LCD related tasks, such as to clear the LCD screen, position the cursor, or to display a character at the current cursor position. Before using the LCD, you need to initialize it by calling the InitializeLCD routine. For example, if you want to clear the LCD display and write "Hello World!!!" from your VEX application main function, you can use the C code which writes the LCDCLS command directly to the Hitachi LCD controller. Be sure to include the "#include "lcd.h" file so that the LCD functions become available; also include lcd.c in your project.

TABLE 2. Bill of Materials for the LCD display.

ITEM	QTY	DESCRIPTION	SOURCE
1	1	VEX Controller	Innovation First, Inc. **www.vexforum.com**
2	1	16 x 2 LCD Display	SparkFun **www.sparkfun.com**
3	40	.100 Pin Headers	Digi-Key **www.Digi-Key.com**
4	1	10K ohm Trim Potentiometer	Digi-Key **www.Digi-Key.com**
5	1	Wire-wrap Cable	RadioShack **www.radioshack.com**
6	1	Package of Jumper Cables (optional)	RadioShack **www.radioshack.com**

The total quantities of each component depend on how many MAX7219 ICs are used (one or two). If you plan to use point-to-point construction, then there is no need to purchase the wire-wrap materials.

```
// Setup timers, configure ports and initialize
// variables
SetupTimers();

// Initialize the LCD Display
InitializeLCD();

// Send the LCD Clear Display command to the LCD
// and have it point to
// row 1, column 1.
WriteLCDCommand(LCDCLS);

// Use the WriteLCD function to write individual
// characters starting from the
// current cursor position as follows:
WriteLCD('H');
WriteLCD('e');
WriteLCD('l');
WriteLCD('l');
WriteLCD('o');

// Wait 5 second between words for pacing
pause(5000);

// You can also use my LCD version of the printf
// function to direct the text
// output to the LCD Display:
printf("World!!!\r");

// The '\r' causes a carriage return to the next
// line on the LCD Display.

printf("This is a test");
```

LISTING 1. Some of the details regarding using the LCD display driver which was written in PIC8 C using MPLAB.

```
    // Setup timers, configure ports and
    // initialize variables
    SetupTimers();

    // Initialize the LCD Display
    InitializeLCD();

    // This test will display integer values
    // from 0 to 255
    while (1)    // Forever
    {
            for (i=0; i<256; i++)
            {
                    printf("Testing %d
times!!!\n", i);
                    pause(500);

            }
     }
```

LISTING 2. An example of how to print integer values or hex values to the LCD display.

Connecting the LCD to the VEX Controller

From the schematic shown in **Figure 3**, you see how simple it is to connect the LCD to the microcontroller's digital output pins. Just solder .100 pin headers to the LCD module to wire-wrap or jumper the connections. Another connection alternative is to use the pin headers with a ribbon cable for a more rugged display.

The LCD obtains its power directly from the microcontroller by connecting the VSS pin to one of its ground pins, and by connecting the VCC pin to one of the microcontroller's 5V power supply pins (also shown in the **schematic**). The only external component (other than the LCD module) required is a 10K ohm trim potentiometer which is used to control the LCD contrast function. Use it to insure that text is visible on the display under various lighting conditions (indoor, outdoor sunlight). Some newer displays have two extra pins (power and ground) for an LCD backlight. If this is the case, then use the manufacturer's recommended voltages for the backlight.

Bill of Materials

The parts required to build the LCD display circuit are shown in **Table 2**. Other than the microcontroller, the other components are relatively low cost and can be found at your local RadioShack, or on the Internet at **Jameco.com** or SparkFun. The optional jumper cables make it very easy to connect an LCD to the controller, but should only be used for temporary connections since they can easily pull out of the controller.

LCD Display Firmware

If you want to run a simple test of the LCD to make sure that it works, then download the lcd.hex file using the IFI bootloader. Once you have verified the correct operation of the LCD, you can start using it for your own VEX applications or to customize the drivers. The PIC18 C code that drives the LCD is located in lcd.c. Included in this file are functions to position the cursor, display text characters at a specific line and column, and display text strings.

The LCD controller test is written in PIC8 C using MPLAB as shown in **Listing 1**. The test demonstrates how to write text to one or both of the LCD lines. I also show you how to position the LCD cursor to a specified row and column, print a character at the specified location, clear the LCD screen, and backspace (shown in **Listing 1**).

In addition to being able to write text integer

FIGURE 4. Another character LCD display that I connected to the VEX controller which is a surplus two line by 24 character display that I purchased online for around $10. The two lines and 24 character line length allow you to display even more information including floating point values and more detailed text messages.

numbers, hex numbers and binary numbers can also be printed on the LCD display, but floating point numbers are currently not supported. The reason you can't print floats using this technique is because the printf routine included with the free PIC18 C Student Compiler does not support floating point format.

The LCD driver code was developed using Microchip MPLAB and the PIC18 C compiler, and also uses IFI libraries. The code configures the VEX controller's digital ports to outputs and initializes the Hitachi LCD controller. Once compiled, the lcd.hex file is generated that is used to program the VEX controller. (I have provided the lcd.hex file so that it can be used without having to compile it.) You'll need the orange programming cable to download the firmware to the controller just like last time.

To include LCD support for your own applications, just add "#include lcd.h" to your C programs and lcd.c to your MPLAB project files.

Once you have downloaded the lcd.hex application using the IFI bootloader, the message "Hello World!!!" should appear. If it doesn't work the first time, check the wiring and the power supply using a digital voltmeter; check for short circuits. Also make sure that the contrast is set correctly using the 10K trim pot.

This version of the LCD firmware includes a special LCD version of printf that can be used to display integer and hex values using the example shown in **Listing 2**. It uses the printf function to convert the integer value from binary to a formatted character string and then sends it directly to the LCD. The pause(n) function is used to delay in milliseconds between each printf call to give the user a chance to read the LCD display before it is overwritten by the next printf.

Displaying Floating Point Values

As previously mentioned, displaying 32-bit IEEE567 floating point values on the LCD is currently not supported in the C firmware that I provided, but is a feature that I would like to add in the future for my own VEX applications. It would require software to reformat the floating point numbers to ASCII text strings so that they can be printed on the LCD. In fact, I provided a routine in my previous article to display "fixed point" numbers on the numeric LED display that could easily be modified to display fixed point numbers and to also handle IEEE567 floating point numbers with exponents on higher resolution LCD displays.

Programming the VEX Controller

The next step is to download the lcd.hex application that is provided with this article into the controller and run it. Start by copying the led.hex file to your computer's hard disk and place it in a folder. This is done by running the IFI bootloader and browsing to the directory that you created for the lcd.hex file. Turn the power switch on and download the led.hex file.

Once the lcd.hex file has been downloaded, the LCD should show "Hello World!!!" on the first line of the 2 x 16 character display.

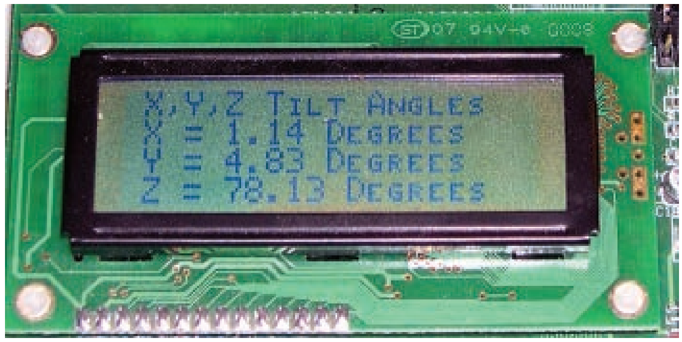


FIGURE 5. This accelerometer display was generated using a Microchip dsPIC30F6014 development board graphics LCD display which uses the SPI interface. Notice that it does not have any problems with floating point when displaying the xyz tilt angles since the Microchip C30 compiler provides the necessary floating point libraries.

Using Other LCD Displays Work with VEX?

Another character display that I connected to the VEX controller is shown in **Figure 4**. It is a surplus two line by 24 character display that I purchased online for around $10. The two lines and 24 character line length allow you to display even more information, including floating point values and extended text messages.

Graphics LCD Display Applications

Applications for new graphics displays are limitless and can be integrated into appliances, medical devices, automotive devices, and other embedded devices. The new QVGA interface provides access to high resolution graphics OLED displays and color LCD displays. Except for the serial RS-232 interface, these interfaces are more involved requiring more capable microcontrollers with built-in I^2C and SPI peripherals.

The accelerometer display shown in **Figure 5** was generated using a dsPIC30F6014 development board LCD display which uses the SPI interface. Notice that it does not have any problems with floating point when displaying the xyz tilt angles since the Microchip C30 compiler provides the necessary floating point libraries.

Going Further

Using the information in this article, you can easily

interface LCD displays to other kinds of microcontrollers, although you may need to port the C code to the selected architecture and map the LCD pins accordingly. The PIC18 series in particular will work by selecting the target processor and recompiling using PIC18 C with only minor changes to the C firmware.

I have shown you how to interface your own low cost LCD to a VEX microcontroller and now it is up to you to put this information to good use in your own applications.

In the next installment, I plan to show how to interface a low cost 4 x 4 keypad to the VEX microcontroller that can be used with the LCD display, so you will be able to build a simple user interface for your VEX projects.

Until VEX time ... **SV**

The Official VEX LCD Display

The new VEX LCD display shown in **Figure A**, measures 2.6" W x 0.63" H and provides 16 characters x two lines. You can use this display to receive real time feedback from your robot to perform live debugging, view multiple stored program configurations and select between them, or provide additional user input to your robot.

There are two interconnect ports on the side of the LCD module. These ports are labeled RX and TX. The RX and TX ports are labeled with respect to the device that connects to the LCD display module. There are three pushbutton switches located on top of the VEX display module. These three pushbutton switches can be configured by the user through software.

FIGURE A. The new VEX LCD display.

Add a Transceiver to Your ZeroG Trainer

By Fred Eady

If you invested some of your precious time and hard-earned money in last month's Trainer Expansion project, you're about to receive a dividend. This month's project will reuse last month's base hardware technology and add a high-powered Microchip 802.15.4 transceiver to the mix.

What's Different?

Last time, we took a cue from *Nuts & Volts Magazine* and replaced the ZeroG - PIC24FJ128GA006 Trainer's original PIC24FJ128GA006 with a 32-bit PIC32MX795F512H. We also lost the ZeroG Wi-Fi module and electrically attached an XBee-Pro 802.15.4 transceiver to our electronic Frankenstein via a garage-brewed perfboard carrier assembly. While we were slinging solder, an SP3232-based RS-232 portal was installed along with a Microchip Explorer 16 vintage LCD.

This Trainer project will again take a cue from the Design Cycle column over in *Nuts & Volts*. However, the *SERVO* version is sans USB. Otherwise, the PIC24FJ/PIC32MX Trainer printed circuit board (PCB) basking in the lights in **Photo 1** is virtually identical to the Design Cycle PIC24FJ/PIC32MX USB Trainer PCB showing its teeth in **Photo 2**. The only other difference worth noting is the location of the voltage and ground feeds that provide a power path to the auxiliary board.

Can This Be Done?

The peripheral that I would like to add to our existing Trainer is a Microchip MRF24J40MB IEEE 802.15.4 2.4 GHz transceiver. The MRF24J40MB is the higher powered version of the MRF24J40MA.

There are no preexisting PIC24FJ128GA006 hardware

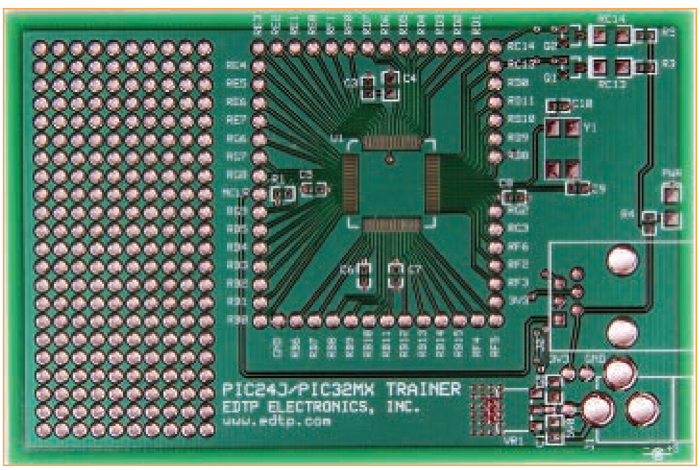


PHOTO 1. The PIC24FJ/PIC32MX Trainer is designed to accommodate any 64-pin PIC that is pin-compatible with the PIC24FJ128GA006. Eligible microcontrollers include the 16-bit USB-capable PIC24FJ256GB106 and the multi-talented 32-bit PIC32MX795F512H.

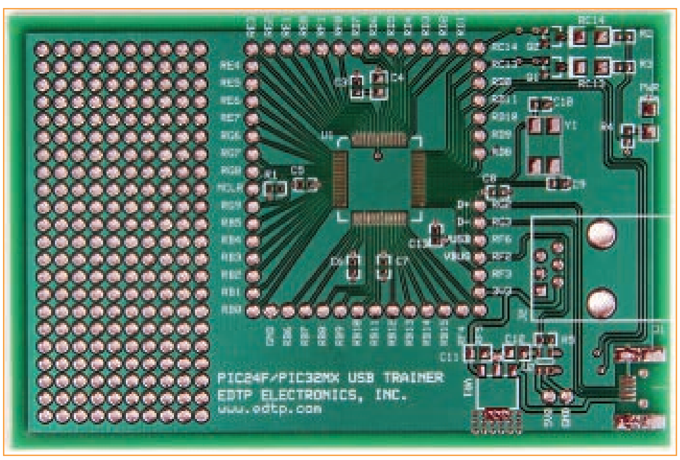


PHOTO 2. The PIC24FJ/PIC32MX USB Trainer is the USB-enabled variant of the Trainer. This PCB is designed to host any microcontroller that is pin-compatible with a PIC24FJ256GB106 or PIC32MX795F512H.

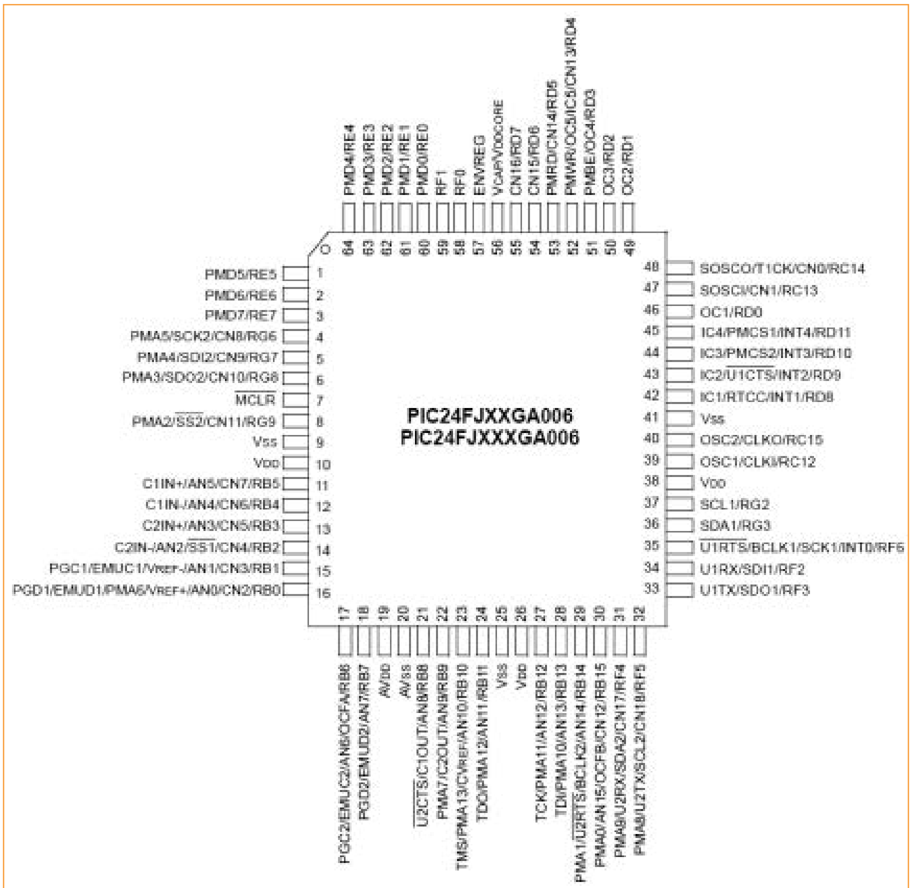


FIGURE 1. The expansion base we built earlier utilizes the U2TX and U2RX I/O pins for the RS-232 interface. That leaves the SPI0 portal I/O pins which are shared with UART1 — clear and available for our MRF24J40MB.

What's a MiWi?

MiWi is a simple wireless networking protocol designed for low data rate, short distance, low cost networks.

or MiWi (see **Sidebar**) templates to work from. So, we'll have to do some investigative work to determine if we can actually drive an MRF24J40MB using this PIC. First, let's try to attack the problem from the hardware perspective.

A quick scan of the MRF24J40MB datasheet tells us that we need to feed it with a 3.3 volt power supply that can supply a minimum of 130 mA. No sweat as a 500 mA 3.3 volt regulated power source is part of the base Trainer system. The MRF24J40MB's data I/O subsystem consists of a standard four-wire SPI portal. I know I don't normally read magazine articles and datasheet simultaneously (and you probably don't either), so I've posted **Figure 1** for your convenience. A trip around the PIC24FJ128GA006 graphic you see in **Figure 1** reveals a pair of four-wire SPI portals. Since every member of the PIC24FJ128GA006 family has the maximum of a pair of SPI portals, we can get up on our donkey and conclude that we have the necessary SPI resources to move data between an MRF24J40MB and our PIC.

Going with the positive educated guess about the power of our PIC24FJ128GA006's SPI engines, let's go one step further and prove our conclusion. The MRF24J40MB can be had in a PICtail Plus package. That means there's most likely a schematic diagram we can get some ideas from in the MRF24J40MB PICtail Plus User Guide. Our donkey ride pays off in **Figure 2**. The MRF24J40MB's active-low CS (Chip Select) line designation matches up perfectly with the PIC24FJ128GA006. Likewise for the MRF24J40MB's SCK (SPI Clock) RF6 pin assignment. Then, it seems that the PIC24FJ128GA006 runs out of PORTF pins. No worries! We'll simply modify the SDI1 and SDO1 definitions in the MiWi firmware. The active-low RESET line looks to be on the spot as does the WAKE trigger line. There is no RE8 I/O pin shown in **Figure 1**. So, we'll look at the function that RE8 provides and match up that function with the appropriate PIC24FJ128GA006 I/O pin. In this case, we'll redefine RE8 as RD8 in the MiWi hardware definitions. You can get down off that donkey as we have positive proof that this project can be realized.

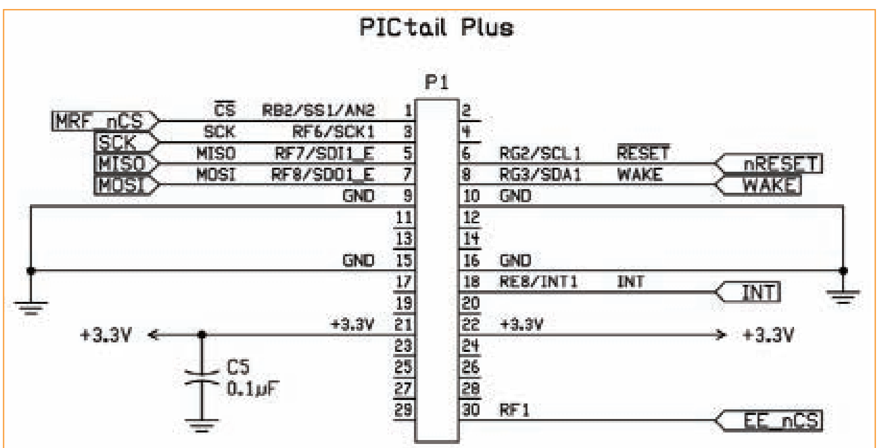


FIGURE 2. When you're unsure about how to wire up a part, PICtail Plus schematics are of great value as they normally outline the use of a particular part for multiple microcontrollers and demo boards.

MRF24J40MB 101

The MRF24J40MB transceiver you see in **Photo 3** is the long bow version of this transceiver. The MRF24J40MA transmits at 0 dBm which equates to one milliwatt. The MRF24J40MB can emit a +20 dBm or 100

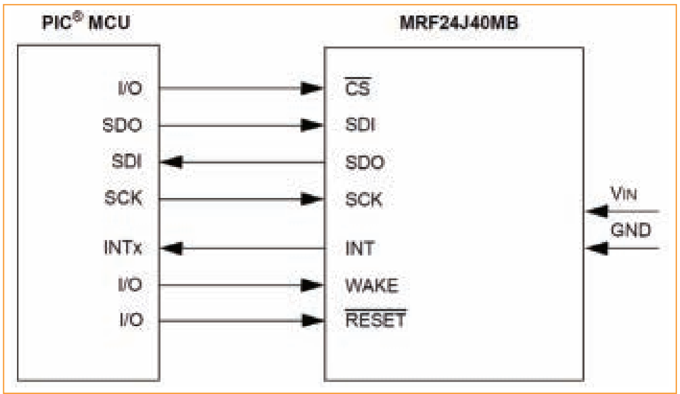


FIGURE 3. The MRF24J40MB can stuff up to 250 kbps of data through an 802.15.4 link. A typical SPI portal can run as fast as its MASTER node can reliably clock data in and out of it.

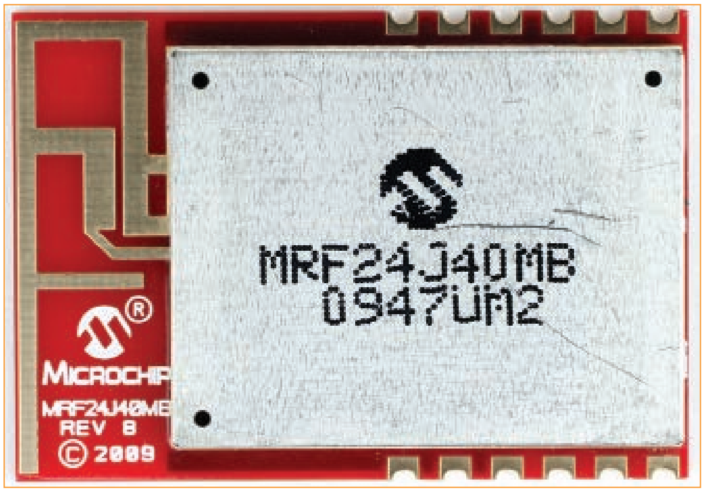


PHOTO 3. The MRF24J40MB is easily mounted SMT style or via pins and headers.

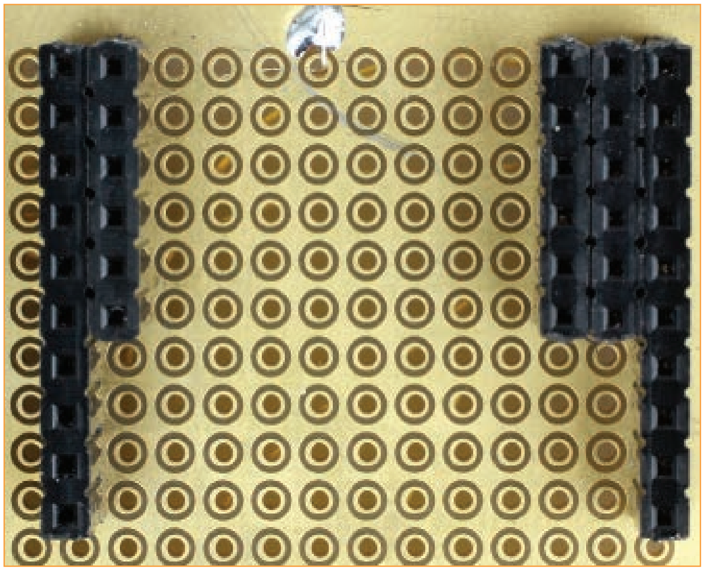


PHOTO 4. As Spock would say, "Random chance seems to have operated in our favor." There is NO WAY that I purposely designed these socket areas to overlap so perfectly.

milliwatt signal. The higher transmit power rating allows it to push a signal out to a maximum of 4,000 feet with no need for a license. The MRF24J40MB construction includes a PIC24FJ128GA006 (power amplifier) and LNA (low noise amplifier) that allows it to outperform the MRF24J40MA on both the transmit and receive levels. If you want to know all of the pointy hat details, I'll leave reading the datasheet up to you. I'm interested in moving some data using the Earth's magnetic field and as you're reading this you are most likely in that mindset, as well. So, let's only look at the physics of the MRF24J40MB that interest us. For instance, what do all of its pins do?

The active-low RESET pin performs the obvious function of a global reset input. The WAKE input implies that the MRF24J40MB can be put to sleep to conserve power. Most microcontrollers have ample time to poll external communications resources like the MRF24J40MB. However, it's always a good thing when the microcontroller can go about its business and only service a device when required to do so. That's where the INT output pin comes into play.

The INT pin is an output that is tied to one of the host microcontroller's external interrupt inputs. When the MRF24J40MB needs to perform data I/O or do something the host microcontroller needs to know about, the MRF24J40MB activates its INT pin. The activation of the INT output forces the host microcontroller to run an interrupt handler routine to service the INT-initiated request. The four-wire SPI portal is comprised of the SDO, SDI, SCK, and CS I/O pins. Rather than ramble through how all of these MRF24J40MB signals hook up to a host microcontroller, a quick perusal of **Figure 3** makes it perfectly clear.

Installing an MRF24J40MB

Normally, we would design and fabricate a specialized PCB for this project. However, our initial inclusion of the high-quality Twin Industries breadboard in the Trainer Expansion project allows us to mount and connect our MRF24J40MB just as its datasheet recommends. The MRF24J40MB is designed to be mounted over a ground plane with its printed circuit antenna hanging off the edge of the host PCB. The Twin Industries 8100-45-LF breadboard is equipped with plated through holes and a single copper plane. We will ground the breadboard plane to meet the MRF24J40MB installation requirements.

As you can see in **Photo 4**, the MRF24J40MB footprint puts its mounting pads just inside of the shade-tree engineered XBee-Pro mounts. The actual electrical hookups are shown in **Schematic 1**.

Customizing the MiWi Stack

Building a MiWi definition file (MiWiDefs.h) is as easy as starting up a ZENA MiWi stack configuration session. Basically, the stack configuration utility wants to know what demo board or microcontroller you will be using, how fast it will be clocked, and what microcontroller I/O and interrupt pins you would like to assign to the MRF24J40MB

connection. You can also specify a unique MAC address and security options. Once you've entered all of the required information, the ZENA stack configuration utility generates a MiWiDefs.h file based on your entries. The proof in the pudding is shown in this code snippet which reflects the MAC address I entered:

```
// MAC Address
#define EUI_7 0x00
#define EUI_6 0x04
#define EUI_5 0xA3
#define EUI_4 0x11
#define EUI_3 0x22
#define EUI_2 0x33
#define EUI_1 0x44
#define EUI_0 0x55
```

We also know from experience that the PIC24FJ/PIC32MX Trainer clocks its CPU at 32 MHz:

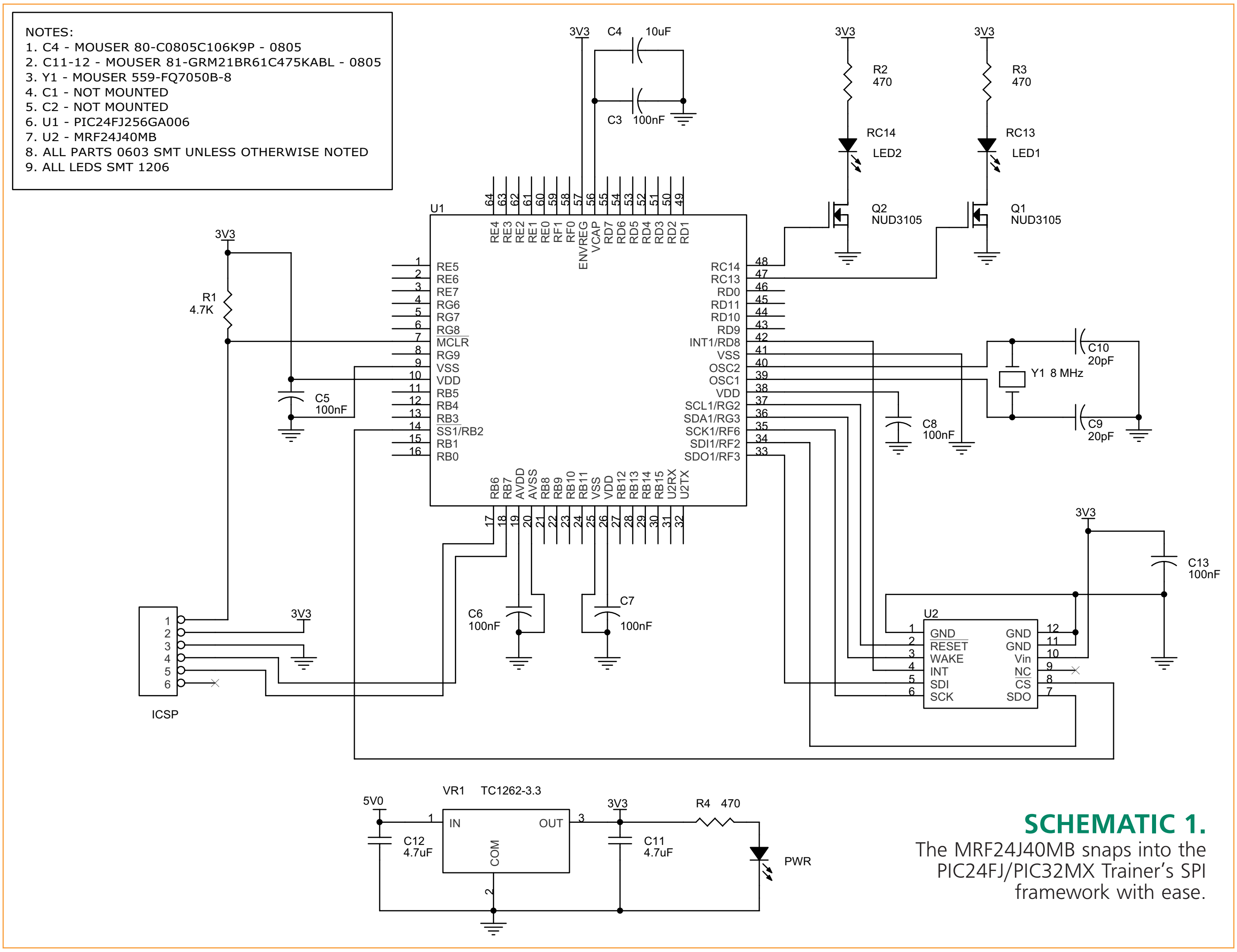
```
// PIC Information
#define CLOCK_FREQ 32000000
#define BAUD_RATE 19200
```

I specified the baud rate during the MiWi stack configuration process.

The Trainer will be programmed to be a PAN coordinator, or Personal Area Network boss. In addition to being coordinator-capable, I configured our Trainer as an FFD (Full Function Device). Here's what was generated in the MiWiDefs.h code:

```
// Device Information
#define I_AM_FFD
#define DEVICE_TYPE 1          // FFD
#define I_AM_COORDINATOR_CAPABLE
#define ALTERNATE_PAN_COORDINATOR 1
#define RX_ON_WHEN_IDLE 1
#define POWER_SOURCE 1         // Mains
#define ALLOCATE_ADDRESS 1
```

As you've probably already ascertained, a "1" means I am what I say I am. If you've been privy to our previous TCP/IP stack discussions, you know that the aforementioned ASCII definitions are used by various modules of the stack to make decisions as to what to code to compile or which logical branch to take. In a nutshell, our PIC24FJ/PIC32MX Trainer is a coordinator-capable, full function device that is powered by something other than batteries, and it has an ear to the ground at all times. The ASCII and Boolean statements are put into action with the



SCHEMATIC 1.
The MRF24J40MB snaps into the PIC24FJ/PIC32MX Trainer's SPI framework with ease.

assignment of Capability Information bits:

```
#define CAP_INFO ( (((BYTE)ALLOCATE_ADDRESS)<<7)
| (((BYTE)SECURITY_CAPABLE)<<6) |
(((BYTE)RX_ON_WHEN_IDLE)<<3) |
(((BYTE)POWER_SOURCE)<<2) |
(((BYTE)DEVICE_TYPE)<<1) |
ALTERNATE_PAN_COORDINATOR ) // 0x8F
```

Although the bit movement instructions imply that the PIC24FJ/PIC32MX Trainer can be secured, I chose not to activate the security features:

```
// Security Configuration
#define SECURITY_CAPABLE 0
```

Earlier, we had to theoretically prove whether or not we could actually pull this project off. In that process, we successfully assigned all of the necessary PIC24FJ128GA006 I/O pins to the required MRF24J40MB I/O pins to make a decision to move on with the project. I entered those I/O pin assignments in the ZENA stack configuration utility and it produced the following code:

```
// Transceiver Configuration
#define TMRL TMR2
#define RFIF IFS1bits.INT1IF
#define RFIE IEC1bits.INT1IE
#define RF_INT_PIN PORTDbits.RD8
#define PHY_CS LATBbits.LATB2
#define PHY_CS_TRIS TRISBbits.TRISB2
#define PHY_RESETn LATGbits.LATG2
#define PHY_RESETn_TRIS TRISGbits.TRISG2
#define PHY_WAKE LATGbits.LATG3
#define PHY_WAKE_TRIS TRISGbits.TRISG3
```

The utility actually returned bits for external interrupt INT2. However, we actually tied the MRF24J40MB INT pin to the PIC24FJ128GA006's INT1 pin (RD8). So, I overrode the INT2 RFIF and RFIE entries. The remainder of the physical I/O entries match up with the design we've laid down in **Schematic 1**.

Just in case we decide to add a couple of pushbuttons and use the Trainer's LEDs as the PICDEMZ demo board's *main.c* demo application would, I chose to include the pushbutton code from the demo in our code:

```
#define PUSH_BUTTON_1 PORTBbits.RB5
#define PUSH_BUTTON_2 PORTBbits.RB4
#define LED_1 PORTCbits.RC13
#define LED_2 PORTCbits.RC14

#define PUSH_BUTTON_1_TRIS TRISBbits.TRISB5
#define PUSH_BUTTON_2_TRIS TRISBbits.TRISB4
#define LED_1_TRIS TRISCbits.TRISC13
#define LED_2_TRIS TRISCbits.TRISC14
```

Let's check our work.

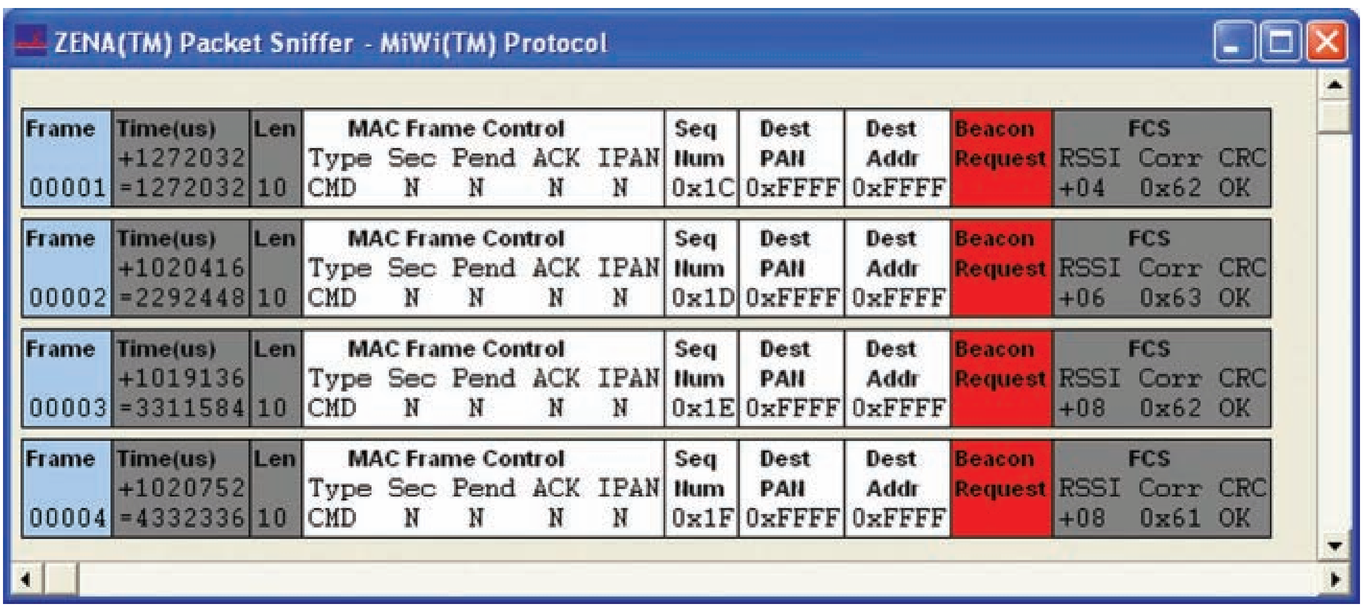
Guess What?

It doesn't work!

So, it's time to pull out some tools and see if we can figure out what is wrong. The known good node is a PICDEMZ End Device node we used in an earlier MRF24J40MA discussion. According to the ZENA traces, the End Device is performing its tasks in a normal manner. With all RF activity shut down, I fired up the End Device and it immediately emitted the Beacon Request frames I captured in **Screenshot 1**.

I shut down the End Device and prepared the PIC24FJ/PIC32MX Trainer MiWi Coordinator node for some air time. I brought up the MiWi Coordinator first and then reapplied power to the PICDEMZ End Device. As you can see in **Screenshot 2**, everything seemed to work as designed. The coordinator and the End Device exchanged niceties and decided they could coexist on the same PAN.

At this point, we should be able to blink the Trainer's LED 2 which is attached to RC14. No such luck. I pressed RB4 on the End Device and saw a packet go out but no joy



SCREENSHOT 1. The MiWi End Device is looking for love in all the possible places. Ideally, a MiWi coordinator will hear this song.

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Dest Addr	Beacon Request	FCS RSSI	Corr	CRC
00001	+33469088 =33469088	10	CMD	N	N	N	N	0x0E	0xFFFF	0xFFFF		-20	0x64	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Source PAN	Source Addr	SuperFrame Specification BO	SO	CAP	Batt	Coord	Assoc	GTS Specification Permit	Count	PendAddr Spec ExtAddr	ShortAddr	Beacon Payload ProtID	Version	Coords	FCS RSSI	Corr	CRC
00002	+5056 =33474144	16	BCN	N	N	N	N	0xA5	0x0077	0x0000	None	None	0xF	N	Y	Y	N	0x0	0x0	0x0	0x4D	0x10	00000001	+00	0x63	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Dest Addr	Source PAN	Source Address	Association Request Alloc	Sec	RxOn	Power	Dev	AltCoord	FCS RSSI	Corr	CRC
00003	+4054608 =37520752	21	CMD	N	N	Y	N	0x10	0x0077	0x0000	0xFFFF	0x0004A31122334455	Y	N	On	Mains	FFD	Y	-25	0x63	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	FCS RSSI	Corr	CRC
00004	+1312 =37530064	5	ACK	N	N	N	N	0x10	-10	0x60	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Dest Addr	Source Address	Data Request	FCS RSSI	Corr	CRC
00005	+1963712 =39493776	18	CMD	N	N	Y	Y	0x11	0x0077	0x0000	0x0004A31122334455		-24	0x63	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	FCS RSSI	Corr	CRC
00006	+1152 =39494928	5	ACK	N	Y	N	N	0x11	-09	0x63	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Destination Address	Source PAN	Source Address	Association Response Status	Address	FCS RSSI	Corr	CRC
00007	+6040 =39500976	29	CMD	N	N	Y	N	0xA6	0x0077	0x0004A31122334455	0x0077	0x0004A31234567893	Success	0x0100	-09	0x66	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	FCS RSSI	Corr	CRC
00008	+1728 =39502704	5	ACK	N	N	N	N	0xA6	-23	0x64	OK

SCREENSHOT 2. As Def Leppard would say, "Fo Fo Fo Fo Foolin."

with LED 2 — which remained dark. With the hardware on the PICDEMZ and PIC24FJ/PIC32MX Trainer side seeming to be attempting to send and receive packets, I turned my attention back to the MiWi *main.c* function.

The first bug I squashed was lurking under a log in the *ConfigFuses.c* file. Recall that I specified a 32 MHz CPU clock to the stack configuration utility. Well, in reality, the MiWi firmware was told that the PIC was running at 8 MHz:

```
#if defined(__PIC24F__)
// PIC24FJ/PIC32MX Trainer
_CONFIG2(FNOSC_PRI & POSCMOD_XT)
       // Primary XT OSC no PLL
_CONFIG1(JTAGEN_OFF & FWDTEN_)
       // JTAG off, watchdog timer off
#endif
```

With no 4x PLL specified in the *ConfigFuses.c* code, the CPU fuse is blown to disable the PLL and the CPU is indeed clocking at 8 MHz. A few clicks on the keyboard will fix that:

```
#if defined(__PIC24F__)
// PIC24FJ/PIC32MX Trainer
_CONFIG2(FNOSC_PRIPLL & POSCMOD_HS)
       // Primary HS OSC with 4x PLL
_CONFIG1(JTAGEN_OFF & FWDTEN_OFF)
       // JTAG off, watchdog timer off
#endif
```

After a quick recompile and program operation, things perked up considerably. The PIC24FJ/PIC32MX Trainer took on a whole new attitude and began to respond without having to be prodded and reset. However, I still could not illuminate the Trainer's LED 2 from the End Device.

Along the way, I noticed that the coordinator's RS-232 messages were not being sent. That told me that the payload bytes that toggle the LED were never being processed. In most cases, that points to an addressing problem. During the time I spent sweating over the *main.c* code line by line, I recall seeing an address string that was also appearing in the unsuccessful data packets that were flowing in the ZENA captures. Here's the *main.c* code snippet I'm referring to:

```
// if no socket, send report by long or short
// address alternatively
if( (transmitMode++ % 2) == 0 )
{
       tempLongAddress[0] = 0x07;
       tempLongAddress[1] = 0x06;
       tempLongAddress[2] = 0x05;
       tempLongAddress[3] = 0x04;
       tempLongAddress[4] = 0x03;
       tempLongAddress[5] = 0x02;
       tempLongAddress[6] = 0x01;
       tempLongAddress[7] = 0x55;

       SendReportByLongAddress(tempLongAddress);
       ConsolePutROMString((ROM char*)"Send
       Report by Long Address\r\n");
}
```

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Destination Address	Source PAN	Source Address	Association Response Status	Address	FCS RSSI	Corr	CRC
00011	+5072 =20769072	29	CMD	N	N	Y	N	0x0B	0x010A	0x5501020304050607	0x010A	0x0004A31234567893	Success	0x0100	+04	0x64	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	FCS RSSI	Corr	CRC
00012	+1744 =20770816	5	ACK	N	N	N	N	0x0B	-15	0x65	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	Dest PAN	Dest Addr	Source Addr	Hops	Frame Control ACK	INTRA	SEC	Dest PAN	Dest Addr	Source PAN	Source Addr	Seq Num	Report Type	ID	Data	FCS RSSI	Corr	CRC
00013	+67753232 =88524048	25	DATA	N	N	Y	Y	0x0C	0x010A	0x0100	0x0000	0x04	N	Y	N	0x010A	0x0100	0x010A	0x0000	0xB6	0x12	0x34	0x55	+03	0x62	OK

Frame	Time(us)	Len	MAC Frame Control Type	Sec	Pend	ACK	IPAN	Seq Num	FCS RSSI	Corr	CRC
00014	+1520 =88525568	5	ACK	N	N	N	N	0x0C	-15	0x63	OK

SCREENSHOT 3. Success! The LED illuminated. The new address is embedded in the MiWi demo firmware and that's what the application is looking for.

With nothing to lose, I performed this little code mod in the MiWiDefs.h file:

```
// MAC Address
//#define EUI_7 0x00
//#define EUI_6 0x04
//#define EUI_5 0xA3
//#define EUI_4 0x11
//#define EUI_3 0x22
//#define EUI_2 0x33
//#define EUI_1 0x44
//#define EUI_0 0x55

#define EUI_7 0x55
#define EUI_6 0x01
#define EUI_5 0x02
#define EUI_4 0x03
#define EUI_3 0x04
#define EUI_2 0x05
#define EUI_1 0x06
#define EUI_0 0x07
```

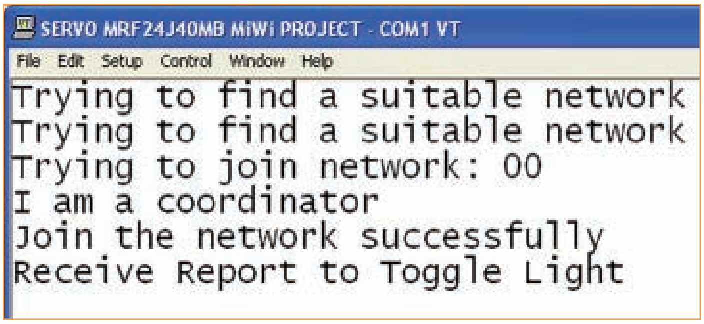
AHA! Take a look at **Screenshot 3**. The association of the coordinator and End Device went as planned with the new address information. So, I pressed the RB4 button on the End Device and the Trainer's LED 2 illuminated. The LED code is only interested in the Report and Data fields of the DATA frame in **Screenshot 3**. Here's the LED blinker code that we were failing to execute in *main.c*:

```
switch(*pRxData++)     //report type
{
  case USER_REPORT_TYPE:
   switch(*pRxData++)     //report id
   {
       case LIGHT_REPORT:
        switch(*pRxData++)     //first byte of
                              //payload
        {
            case LIGHT_ON:
                LED_2 = 1;
                break;
            case LIGHT_OFF:
                LED_2 = 0;
                break;
            case
LIGHT_TOGGLE:
                LED_2 ^= 1;

ConsolePutROMString((ROM

char*)"Receive Report to
Toggle

Light\r\n");
                break;
        }
        break;
   }
   break;
```



SCREENSHOT 4. The presence of the received report message and the LED illumination confirms that the Coordinator LED code was actually executed.

The LED code is a simple parser that switches on the Report Type (0x12) and falls through to switch on the Report ID (0x34) which falls through to the LED LIGHT TOGGLE function (0x55).

The confusion was centered on not reprogramming the PICDEMZ microcontroller. The PICDEMZ is programmed to send to the 0x55, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, and 0x07 address. So, when we changed our ZENA-assigned address on the coordinator, things worked as the coordinator's new address was what the End Device was aiming for. So, guess what? The PIC24FJ/PIC32MX Trainer/ MRF24J40MB combination works! **SV**

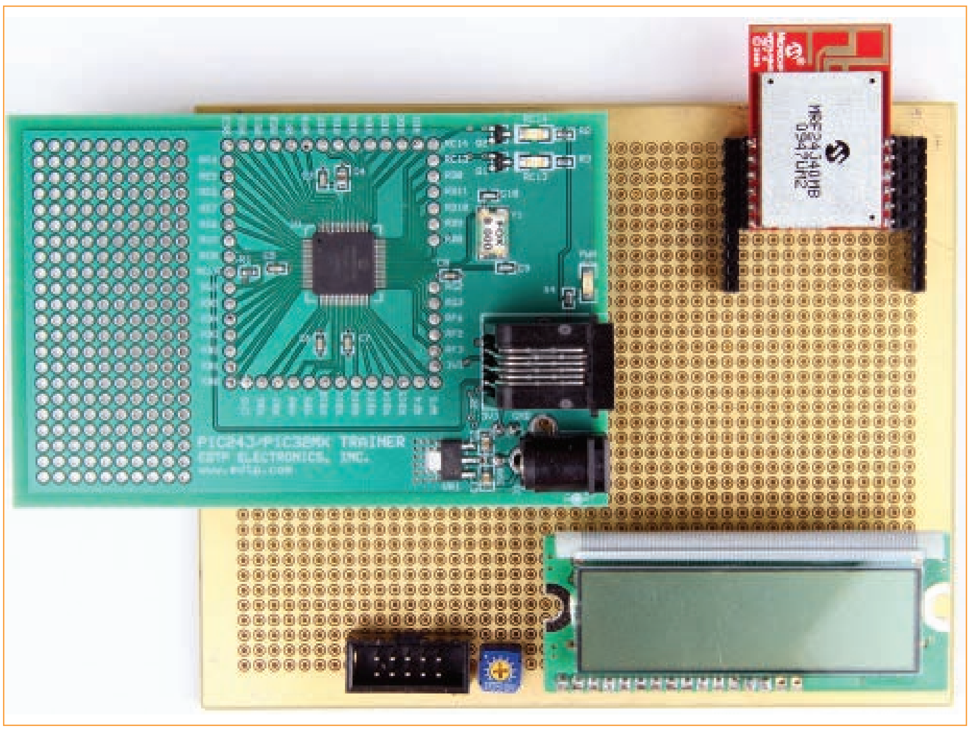


PHOTO 5. Although the XBee-Pro and MRF24J40MB are mutually exclusive due to the interleaved socket arrangement, we're quickly running out of peripheral space on the perfboard.

Sources

EDTP Electronics, Inc.
PIC24FJ/PIC32MX Trainer Kit
ZeroG - PIC24FJ128GA006
Trainer Kit
www.edtp.com

Microchip
MRF24J40MB
MRF24J40MA
MiWi
www.microchip.com

Fred Eady can be reached via email: fred@edtp.com.

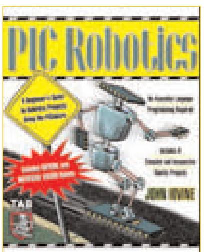
The *SERVO* Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

ROBOTICS

PIC Robotics
by John Iovine

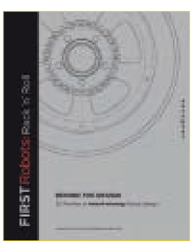


Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.
$24.95

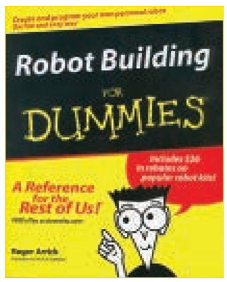
FIRST Robots: Rack 'N' Roll: Behind the Design
by Vince Wilczynski, Stephanie Slezycki

More than 750 photographs!



The second annual book highlighting the creativity and process behind 30 winning robot designs from the 18th annual international FIRST Robotics Competition. The FIRST organization, founded by Dean Kamen (inventor of the Segway), promotes education in the sciences, technology, and engineering.
Reg $39.95

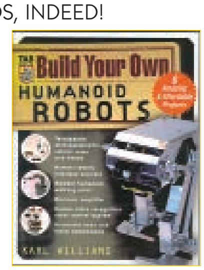
Robot Building for Dummies
by Roger Arrick / Nancy Stevenson



Discover what robots can do and how they work. Find out how to build your own robot and program it to perform tasks. Ready to enter the robot world? This book is your passport! It walks you through building your very own little metal assistant from a kit, dressing it up, giving it a brain, programming it to do things, even making it talk. Along the way, you'll gather some tidbits about robot history, enthusiasts' groups, and more.
$24.95

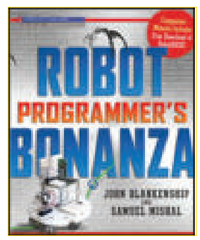
Build Your Own Humanoid Robots
by Karl Williams

GREAT 'DROIDS, INDEED!



This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot. **$24.95***

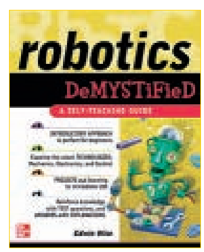
Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.
$29.95

Robotics Demystified
by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!

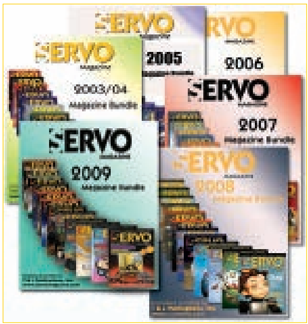


Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles



Published by T & L Publications, Inc.

$57 per bundle

Save $10 off the normal price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Kickin' Bot

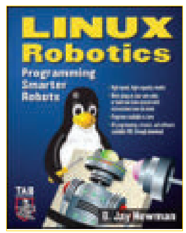
by Grant Imahara

Enter the arena of the metal gladiators!

Do you have what it takes to build a battle-ready robot? You do now! Here are the plans, step-by-step directions, and expert advice that will put you in competition — while you have a heck of a lot of fun getting there. Grant Imahara, the creator of the popular BattleBot Deadblow, shares everything he's learned about robot design, tools, and techniques for metal working and the parts you need and where to get them.
$24.95

Linux Robotics

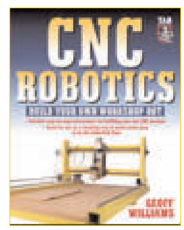
by D. Jay Newman



If you want your robot to have more brains than microcontrollers can deliver — if you want a truly intelligent, high-capability robot — everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.
$34.95

CNC Robotics

by Geoff Williams

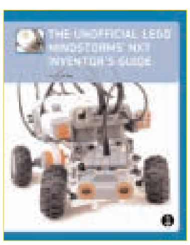


Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting — at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **$34.95**

SPECIAL OFFERS

The Unofficial LEGO MINDSTORMS NXT Inventor's Guide

by David J. Perdue



This book was written for the first version of the NXT set (#8527), and its projects are only compatible with the first version. In other words, because of piece differences between the NXT 1.0 and 2.0 sets, the projects in this book can only be built with an NXT 1.0 set. However, much of the other information is still helpful, and the building, mechanical, and programming details are still applicable.
Reg $29.95
Sale Price $25.95

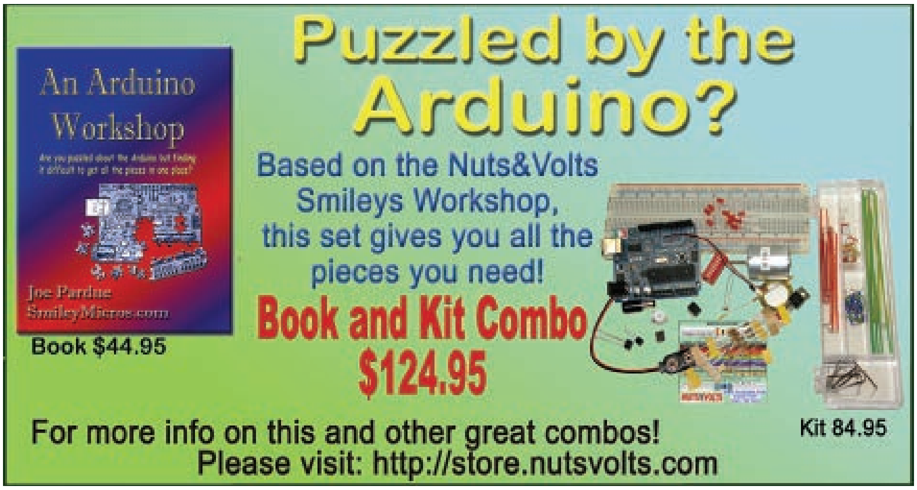
Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

Regular Price $79.95 **Subscriber's Price $75.95**

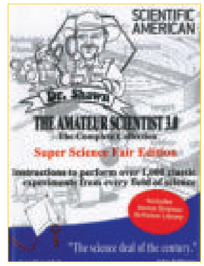
Or order online www.servomagazine.com

SPECIAL OFFERS



The Amateur Scientist 3.0 The Complete Collection

by Bright Science, LLC

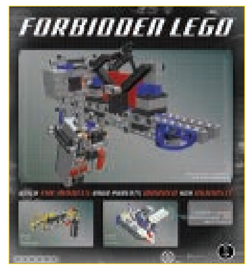


There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do their first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!

Reg $26.95 Sale Price $23.95

Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley



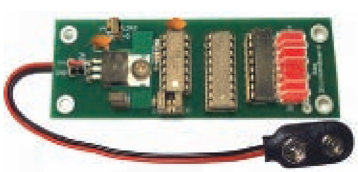
Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg $24.95 Sale Price $19.95**

PROJECTS

Das Blinkenboard Kit

As seen in *Nuts & Volts* June issue
Personal Robotics by Vern Graner
This kit includes a pre-programmed ATtiny84 microcontroller that sports eight software PWM channels to control motor speed and light brightness. Jumper selectable patterns can be used to operate motors, solenoid valves, relays, or any DC load up to 24V/500 mA per channel!

Expand your board with the "Der Magnetfelder Detektor" component pack.



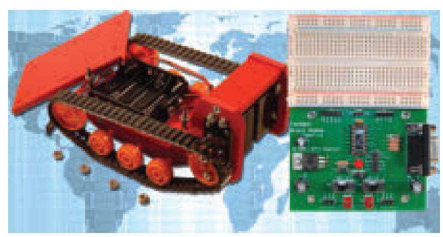
Subscriber's Price **$32.45**
Non-Subscriber's Price **$35.95**
PCBs can be bought separately.

16-Bit Micro Experimenter Board

Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck! In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter.
The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price **$55.95**
Non-Subscriber's Price **$59.95**

Tankbot Kit & Brain Alpha Kit



Tankbot/Brain Alpha originally by Ron Hackett
Now with New Columnist Calvin Turzillo
A series filled with projects and experiments to challenge you through your learning process while you grow your fully expandable Brain Alpha PCB!
The brain is a PICAXE-14A!
For more info & pictures, visit the *SERVO* Webstore.
Tankbot and the Brain Alpha Kit can be purchased separately.

Combo Price $ 138.95

Then and NOW

THE AGE OF ROBOTICS

by Tom Carroll

A lot of people speak of the age of robotics. They've been doing that for decades. Sometimes these words are just hype by a book publisher to promote a book, whereas others are truly convinced that the period in which they live is the beginning of the age of robotics. They see robots everywhere in their lives — on TV, in the movies, cleaning their carpets, exploring space, and building cars world-wide. ***Figure 1*** *from the Japan Robot association illustrates just how fast the robot market is growing. I wanted to capture a few thoughts from others about their ideas on the age of robotics.*

While doing some spring cleaning of my robotics bookcase and file cabinets recently, I saw several titles of articles and books that spoke about how robotics was becoming a part of everyday life. I decided to look at several titles that spanned several decades to see just how thoughts and ideas have changed over the years. Some articles and books spoke glowingly of the "age of robotics" and how robots would soon be scurrying about our homes, serving us as mindless slaves to fulfill our every wish and command. One article that I reread was in the January '07 issue of *Scientific American* that captured the eyes of many robot experimenters, as evidenced by the many online responses from the various robotics groups.

The cover of the magazine (shown in **Figure 2**) boldly stated: "*Dawn of the Age of Robots* Bill Gates writes that every home will soon have smart mobile devices" introduced the article. You can see the brass and steel robot that was interestingly posed to capture the interest of newsstand magazine buyers. The article was titled *A Robot in Every Home* and Gates detractors immediately jumped on his writing, stating that 'this software guy did not know anything about robotics.' Supporters looked a bit deeper and realized that he truly did have insights into home robots, just as he did for home computers 30 years prior. The

FIGURE 2. ***Scientific American*** **— Dawn of the Age of Robots.**

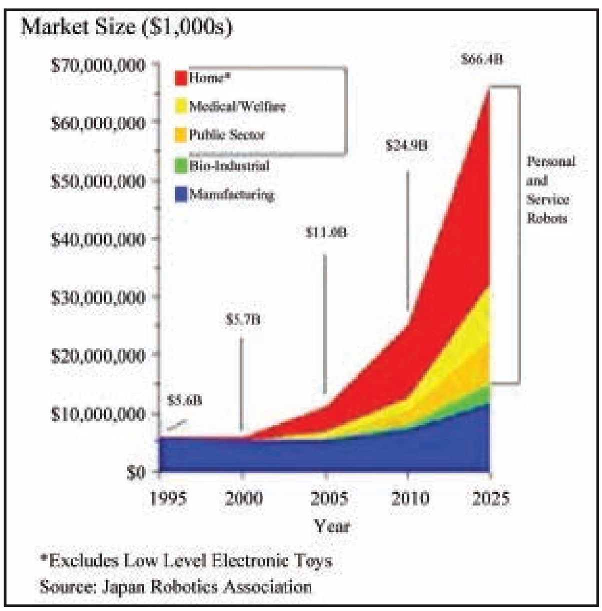


FIGURE 1. Worldwide Robot Market from the Japan Robot Association.

FIGURE 3. American Gothic by Grant Wood; modified by Kenn Brown.

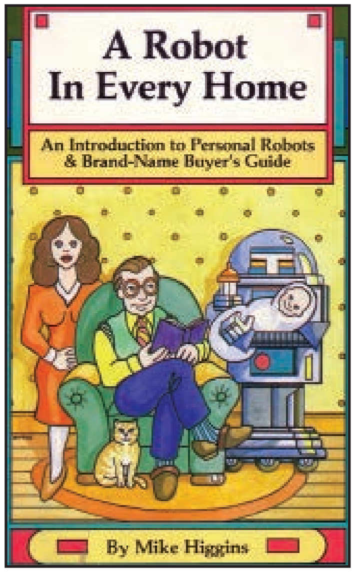


FIGURE 4. *A Robot in Every Home* by Mike Higgins.

accompanying illustration for the article (shown in **Figure 3**) popularly known as *American Gothic* by Grant Wood, was modified by Kenn Brown to exemplify the major impact that robots are making in our homes, with maybe a slight bit of exaggeration.

Gates' article did cover more about the *future* of robotics, but the coverage centered mostly on robots for the home. Some of the artist's drawings depicted several home-specific robots such as an outside mobile security robot, a food and medicine dispensing robot for an elderly person in the home, a Roomba for the floors, and a lawn-mowing robot for the lawns.

Another robot that I found to be rather expensive for the task was a laundry-folding robot mounted to the wall. The two articulated arms and end-effectors would require quite a few axes of motion and a sophisticated vision system well beyond a CMU cam or similar. However, his article did address sensor fusion showing a parallel sensor processor approach versus an approach with sensors in a single serial loop.

So, just what does it take to be in the age of robotics? Is Gates and company riding this wave of robotics? Are we truly there now? Does everyone have to have their own home robot? Will we become complacent in the presence of robots (much like the early scenes of the Will Smith movie, *I-Robot*) where robots walk the streets among people? Will kids in high school soon have 'Robotics' as a required science course? Will there be *Blade Runners* in the future to track down humanoid replicants that have strayed from their assigned areas? Will parents purchase robot babysitters like Robbie from Isaac Asimov's short story series, or an Andrew Martin, like the character portrayed by Robin Williams in *Bicentennial Man*? At what step in our technological evolution will we have entered the age of robotics?

Stepping Back in Time in the Robotics Age

Interface Age — a magazine that supposedly got its start from the Southern California Computer Society's *SCCS Interface Magazine* — published from 1976 to 1982 and was also the first magazine to cover experimental robotics outside the IEEE journals. Some articles in the magazine caught a lot of flack when material was found to be highly exaggerated — especially an article on a promotional robot that supposedly could teach kids French and care for the household. Though this particular article was laughed at by many, it did open the eyes of robotics visionaries.

Back in the late '70s and early '80s, there was a magazine entitled *Robotics Age* (published by Phil Flora) and it was a departure from the several industrial robotics magazines available at that time. In 1981, Carl Helmers became the publisher and in 1983 published a very interesting book entitled *Robotics Age — In the Beginning*. It was basically a compilation of some very interesting articles from the magazine. It was divided into three sections: Power and Control; Interactions: Senses, Vision, and Voice; and Applications and Development. What I found

FIGURE 5. Androbot Bob.

to be so interesting about these articles was how the authors took principles from the industrial sector and made them applicable to home experimenters. The information on sensor fusion, motor control, vision systems using object edge detection, and speech synthesis was written over a quarter century ago, yet, the info is still useful for today's robot experimenters. Published by Hayden Books, I still see it on **Amazon.com** and would highly recommend it to any person who wants to view the complete history of the age of robotics.

Another interesting robot book from the past is *A Robot in Every Home* by Mike Higgins. Published in 1985, the cover (in **Figure 4**) depicts a somewhat whimsical sketch of a robot lover's dream: a family sitting in their living room while a robot is tending the baby. The book was an introduction to the new personal robots that had recently come on the market. It describes the 'big three' of that day: the Androbot TOPO; the HeathKit Hero; and the RB5X from RB Robot. The Androbot BOB — a more sophisticated version of TOPO — is shown in **Figure 5**. These companies made several models and the HeathKit Hero variations were, by far, the best sellers. Higgins had the foresight to realize that this was the first of a new generation of robots when he titled Chapter 7 "Personal Robots: the First Generation."

The Hubot from Hubotics mentioned in Higgins' book was a fairly large robot for those days as can be seen in **Figure 6**. Unfortunately, it did not sell very well as it was little more than a computer monitor/TV on a mobile base with a built-in computer, tape player, and sound system. The overall design of the Hubot seemed a bit ahead of its time, but the look has been applicable to many future designs.

Another book from about 1984 — *The Robot Revolution* by Tom Logsdon — attempted to survey the state of robotics of the mid '80s, yet looked forward to the future of robotics technology. The book's cover (shown in **Figure 7**) depicts visions of how robots can serve us with sketches of robot arms acting as a bartender, a factory worker, and even manipulating the book's title. A very prolific writer on technology, Logsdon was a fellow engineer and friend at Rockwell, and I even helped him with a few chapters. Though I was more involved with robotics at Rockwell, Logsdon had the foresight to assist me in the planning of many of my projects. A true futurist and visionary, his book started with his image of a factory in Japan where robots were building more robots, and ended with some amazing robot technology to serve Mankind. It's a good read for a glimpse of the future of robotics — from 25 years ago.

I ran across another magazine in my organizing that had an interesting article about home robots. In the IEEE *Spectrum* March '86 issue, Glenn Zorpette wrote about *Robots for Fun and Profit*. A quarter century ago, he had interviewed a bunch of robot experimenters around the country about the robots being built. Everyone had different ideas and opinions about the directions robotics was taking. The article steered away from the industrial varieties and centered on experimental robots that operated within private homes, relying on the computing power available to hobbyists of that time.

Back then, home computers were being marketed that had more power than the Apollo moon command module's navigation computer, and all of them cost a few thousand (1980) dollars. Some of us looked a bit into the future at computer-controlling a robot in our homes. We'd seen photos and TV shows about the robot, Shakey, (seen in **Figure 8**) who wandered around Stanford Research Institute's AI Lab from 1966 to the early '70s. This robot could see and plan his way around a room. We just knew we had to build a similar robot of our own. However, Shakey was connected to some rather expensive computers (DEC PDP-10

FIGURE 6. Hubotics Hubot.

FIGURE 7. *The Robot Revolution* by Tom Logsdon.

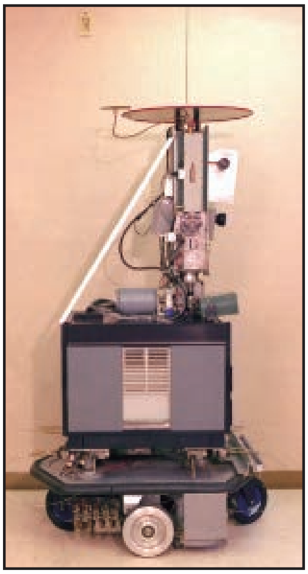


FIGURE 8. Stanford Research Institute's Shakey.

and PDP-15s) that were way beyond the budgets of 1970 home experimenters and robot hobbyists. Some amazing robotic creations began to emerge from the cluttered workshops of home robot experimenters, however, using some relatively powerful microcomputers. Had we actually entered the age of robotics like we thought we had?

Early Personal Computers Powered the First Home Robots

We were convinced that we were on the cutting edge of robotics. We didn't care about those metal monsters in car factories that snaked about car bodies spitting sparks and paint. We had massive computing power right at our fingertips as home computers were becoming popular. A processor running at an amazing 1 MHz, commanding a full 64 kB of DRAM, a five meg hard drive, and 5-1/4" floppy disk drives that we could jam 180 kB of data on — well, the sky was the limit for our soon-to-be intelligent robots.

Long before the days of microcontrollers such as the PIC, BASIC Stamp, and Arduino, we had small, single-board microcomputers such as the John Bell Engineering Model 80-153 to 'talk' to our main desktop computer. This little 3" x 4" card-based computer (shown in **Figure 9**) used the very popular 6502 CPU and had 1K of RAM, 2K of ROM, and a 6522 VIA (versatile interface adapter) for I/O. John Bell also sold an EPROM programmer board to program the 2708 or 2716 EPROMs that held programs.

Now, experimenters could use a computer such as the KIM-1, SYM-1, and the AIM-65 as a larger single-board computer on board, with several smaller 'sub-computers' for sensors, main drive motor control, or appendage control. The Polaroid electro-static ultrasonic range finders from cameras became the eyes for many a robot in those days (as well as these days, three decades later).

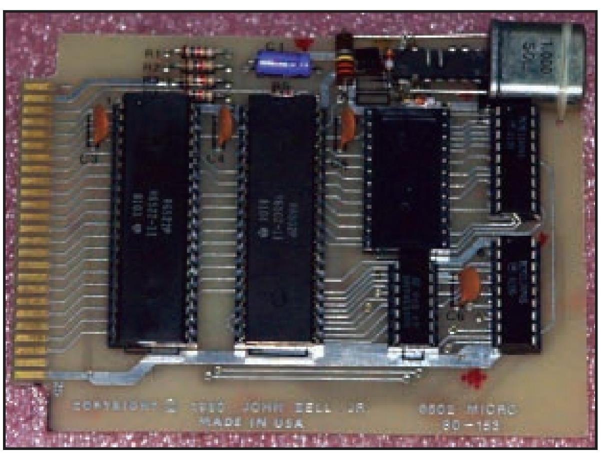


FIGURE 9. John Bell Engineering single-board computer. *(Photo by Don Sawyer.)*

Computer Age Speeds Up Robotics Age

In 1974, Intel debuted the first true microprocessor: the 8080. Few people knew just what to do with it until a small New Mexico company (MITS) built a computer kit — the Altair — that was first publicized in *Popular Electronics* in 1975. Thousands built the kit, and Bill Gates and Paul Allen helped supply the operating system. Another pair of entrepreneurs — Steve Jobs and Steve Wozniak — built and sold the first ready-to-use *personal* computer in 1976: the Apple II. Yes, the computer age was born as people could have useful computers right on their desktop, rather than monstrous main frames hidden in back rooms that only a few could operate.

Robotics continues to claw its way upward. The computer's role for robot intelligence greatly improved things. However, a true robot is more than just a computer on wheels; it is a life-form of sorts created to entertain, enlighten, and teach us.

Final Thoughts

Is the basic, all-in-one house-cleaning robot that dusts shelves full of figurines, cleans windows inside and out, scrubs and sweeps floors, walks the dog, and mows the grass still a dream? Is Isaac Asimov's Robbie still the trusted babysitter of the future? Are the NS-5 robots that tormented Detective Spooner in *I-Robot* what we can look forward to? Is the age of robotics any closer than it was 25 years ago? Mike Budimir wrote an article for *Machine Design* — A Robot in Every House — in 2002, stating: "Robots are not just for assembly lines any more. They're knocking at your front door, ready to fetch you a beer, watch for prowlers, or play your favorite CD." Yes, robots will continue to proliferate in our society, since we all know that the age of robotics is here to stay. **SV**

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

ADVERTISER INDEX


RobotShop
Robotics at your service! ™
Robots, lots of robots!
The World's Leading Source
for Domestic and Professional Robot Technology
www.robotshop.com